COMPATIBILITY CONDITIONS AND APPLICATION TO DAMAGE IN DISCRETE STRUCTURES

by

Predrag Krtolica

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Mathematics

The University of Utah

December 2015

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Predrag Krtolica**
has been approved by the following supervisory committee members:

Andrej Cherkaev and Andrejs Treibergs ,	Chair(s)	**29 Oct 2015** Date Approved
Peter Alfeld ,	Member	**29 Oct 2015** Date Approved
Grzegorz Dzierżanowski ,	Member	**30 Oct 2015** Date Approved
Yekaterina Epshteyn ,	Member	**29 Oct 2015** Date Approved
Seubpong Leelavanichkul ,	Member	**29 Oct 2015** Date Approved

by **Peter Trapa** , Chair/Dean of
the Department/College/School of **Mathematics**
and by **David B. Kieda** , Dean of The Graduate School.

ABSTRACT

We identify the number of compatibility conditions in 2-D and 3-D discrete structures as a measurement device in determining the rigidity of structures and their resilience to damage. Different shortcuts in counting compatibility conditions are developed and proved.

Computing discrete compatibility conditions in hexagonal structures that undergo small enough deformation to be safely linearized, and using these results to say something about the continuum compatibility condition, is the main part of this document. Namely, it is shown that the linearized discrete compatibility condition of hexagonal lattices imposes the 2-D infinitesimal compatibility condition.

The computation of discrete 3-D compatibility conditions is done by studying a cuboctahedron. The results show that, because of its infinitesimal flexibility, the extra degree of freedom in a linearized system produces an extra compatibility condition. However, the breaking of nongeneric symmetry of a cuboctahedron by any nonzero perturbation of its nodes makes it infinitesimally rigid, and the extra compatibility condition is lost.

The role that compatibility conditions play in damaged structures is demonstrated by experiments on hexagonal lattices. The correlation between the loss of compatibility conditions and the spread of damage is made, as well as the importance of strategic placing of stronger and weaker links in an attempt to strengthen a structure with given boundary conditions and loading.

CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

First I would like to thank my co-chairs Andrej Cherkaev and Andrejs Treibergs for their immeasurable help. Only the three of us know how much collective struggle and frustration was involved in this research, and how much patience and hard work was needed to produce this final document. Thank you both for this, though painfully frustrating at times, invaluable experience that has been instrumental in my growth as a mathematician. Additionally, I am thankful to the rest of my committee members for their support, help, and friendship, especially to Peter Alfeld whose knowledge in computer programming proved to be very helpful at a critical time.

I owe a big thank you to so many who work in the Math Department at the University of Utah for their support, endless patience, and friendship that I will cherish for the rest of my life. I have met the most wonderful human beings at this department, who genuinely enjoy helping others succeed. In particular, I consider myself extremely lucky to have had the privilege and opportunity to meet and work with Nelson Beebe, whose attention to details and expertise in LaTeX proved decisive in my thesis write-up. Outside of the department the outstanding help came from my CDS advisor Sid Davis.

But most of all, I am thankful to the most loving and supportive parents and sister a person could ask for, whose unlimited and unconditional love is the only reason any of my accomplishments were possible. It is useless to try and express the gratitude I have for your unlimited help and unparalleled determinism and faith in me.

CHAPTER 1

INTRODUCTION

In the first two chapters the concept of compatibility conditions is introduced, as well as different methods and shortcuts for counting these in various structures. The connection between the number of compatibility conditions and the strength of a structure — the resilience level — is revealed. It is found that, in general, the more compatibility conditions a structure has, the stronger it is, because rigidity *can be* sustained until every single compatibility condition is lost, where the number of compatibility conditions gives the maximum number of links a structure can lose without losing its rigidity.

It is important to emphasize *can be* in the previous sentence because a structure with C_d compatibility conditions *can* lose exactly C_d links and still stay rigid, but not *any* C_d links — most structures can lose rigidity by losing only one or two links, regardless of compatibility conditions.

In addition, knowing the number of compatibility conditions of a structure does not say which links are removable, or, more precisely, redundant. In fact, the algorithm for finding any combination of such links is an extremely difficult task, especially because it depends highly on the geometry of a structure, as well as on fixed parameters. This is briefly attempted in Section 9.3.3 on page 89 with the introduction of damage clusters.

The derivation of continuous compatibility conditions is done in Chapter 5 on page 42, where the alternative method of projections in Fourier space is presented, while the discrete compatibility conditions are introduced in Chapter 6 on page 51, where the wagon-wheel conditions for different hexagons are found. It becomes apparent at that point that the rest

of the document uses hexagonal lattices as the only structures, where the smallest unit of compatibility, which is defined to be the smallest structure in a periodic lattice with at least one compatibility condition, is a hexagon.

The main part of this document is Chapter 7 on page 61, where the connection between the wagon-wheel conditions — compatibility conditions of hexagons — and the 2-D continuous compatibility condition is made. It is shown that 2-D continuums under small deformation obey the same compatibility conditions as hexagons, when taken to a δ-limit. The strains need to be single valued and continuous so they can be approximated by Taylor polynomials, up to some degree r. This enables the studying of compatibility conditions of a hexagonal approximation to a continuum.

Compatibility conditions of 3-D structures, more precisely of a cuboctahedron, are discussed in Chapter 8 on page 68. Similar methods of linear algebra are used to find the four compatibility conditions of a regular cuboctahedron. Analysis shows that these four conditions correspond to the wagon-wheel conditions of each of the four intersecting hexagons that each regular cuboctahedron possesses. It is shown that the breaking of the symmetry via any nonzero perturbation of a cuboctahedron's nodes leads to infinitesimal rigidity, which leads to the loss of the four planar compatibility conditions in place of the three spatial ones.

The mechanics of hexagonal lattices is explored in the last chapter. Many experiments are conducted, such as the strengthening of structures by redistribution of mass, where the design and mass of a structure is kept constant while strengthening the more critical links at the expense of the rest. In addition, the experiments showing the propagation of damage while losing compatibility conditions are conducted, and improvement strategies are suggested. Results from previous sections are used to explain the found phenomena.

CHAPTER 2

BASIC DEFINITIONS

In this chapter, several concepts that are used throughout this document are defined. Most of these definitions are modified or exact versions used first by James Clerk Maxwell in his article from 1864 [23], by Connelly [13, 14], Strang [31], Satyan L. Devadoss and Joseph O'Rourke [15], Pei Chi Chou and Nicholas J. Pagano [9], and Martin H. Sadd [30].

The lectures by Igor Pak [24] are also used, especially when writing definitions about polygons, as well as the works by Ivan Izmestiev [18] when defining frameworks, and, later, infinitesimal deformation.

For the review on rigidity, other than the references already mentioned, a thesis by Mykyta V. Chubynsky [10], Herman Gluck's paper [16], Gerard Laman's paper [21], and the works by R. Connelly [12–14] are also used.

With definitions of compatibility, strains and stresses, the most useful papers happen to be by Andrej Cherkaev, Andrei Kouznetsov, and Alexander Panchenko [7], as well as the lecture notes by P. Kelly [20].

Definition 1 (Configuration)**.** *A set of n labeled and ordered points (nodes), $S(n) = \{a_0, a_1, \ldots, a_{n-1}\}$, where $a_i \in \mathbb{R}^d$, for all $i = 0, \ldots, n-1$, is called a configuration.*

Let $E(m)$ be a set of m pairs of distinct points from a given configuration that are connected by straight 1-D links. Then a *framework* consists of the given configuration, $S(n)$, together with $E(m)$ [14].

Definition 2 (Structure)**.** *Any d-dimensional connected framework is called a structure, and is denoted by $Q^d(S)$.*

Let $Q^d(S)$ be any structure with n frictionless, universal nodes,[1] connected by m links, denoted by $l_{i,j}$, such that $l_{i,j} = a_i - a_j$ is a vector that connects node i with node j. Then

$$L_{i,j} = |l_{i,j}| = |a_j - a_i|\,, \quad i,j \in \{0,\ldots,n-1\} \tag{2.1}$$

is the length of $l_{i,j}$, where $|\cdot|$ is the Euclidean norm, or the distance measured using the Pythagorean Theorem. In fact, for $a_i, a_j \in \mathbb{R}^d$, where $a_i = (a_i[1], a_i[2], \ldots, a_i[d])$ and $a_j = (a_j[1], a_j[2], \ldots, a_j[d])$, it implies that

$$\begin{aligned} |a_j - a_i| &= \sqrt{(a_j[1]-a_i[1])^2 + (a_j[2]-a_i[2])^2 + \cdots + (a_j[d]-a_i[d])^2} \\ &= \sqrt{\sum_{k=1}^{d} (a_j[k]-a_i[k])^2}, \end{aligned} \tag{2.2}$$

where $a_i[j]$ is the j-th component of the i-th node. It follows that $l_{i,j} = -l_{j,i}$, and $L_{i,j} = L_{j,i}$. Then the triangle inequality,

$$L_{i,k} \leq L_{i,j} + L_{j,k} \quad \text{for all } i,j,k = 0,\ldots,n-1, \tag{2.3}$$

holds.

Definition 3 (Flex, Rigid Flex)**.** *A flex of $Q^d(S)$ is a continuous motion of all the nodes $a(t) = (a_0(t), a_1(t), \ldots, a_{n-1}(t))$, $0 \leq t \leq 1$, so that $L_{i,j}(t)$ is constant for all t, and for all $l_{i,j} \in E(m)$. The flex $a(t)$ is* rigid *if all the distances, $|a_k(t) - a_l(t)|$, are constant, for all t and all $k,l \in \{0,1,2,\ldots,n-1\}$. [13]*

Any rigid flex, $a(t)$, is obtained by applying a continuous family of rigid motions to $Q^d(S)$, i.e., to each $a_i(0) = a_i \in S^d(n)$. Thus,

$$a(t) = (g_t a_0(t), g_t a_1(t), \ldots, g_t a_{n-1}(t))\,,$$

where g_t represents rigid motion of $Q^d(S)$ [13].

Definition 4 (Rigidity)**.** *A structure $Q^d(S)$ is* rigid *if every flex of it is rigid. [14]*

---

[1] *Universal, frictionless node* is free to move in any direction [18].

More precisely, a structure is rigid if

$$|a_i(t) - a_j(t)|^2 = (a_i(t) - a_j(t))^T \cdot (a_i(t) - a_j(t)), \tag{2.4}$$

for all $a_i, a_j \in S(n)$ and $0 \leq t \leq 1$. That is, the square distance between *any* two nodes, connected or not, stays the same as time varies.

Informally, a structure is rigid if it cannot be deformed or strained. In other words, rigidity is the absence of *relative* motion in a structure, and a structure is rigid if it is not a mechanism — it does not admit finite motion [28].

Another useful definition: a structure is rigid if the only way to deform it is to change the lengths of the links. A few examples of rigid and nonrigid structures are given in Figure 2.1 on page 7.

On the other hand, structures that do not even admit any infinitesimal motion are called *infinitesimally rigid*.

A useful way of thinking about infinitesimal rigidity is if we allow the lengths of links to vary, and require that the instantaneous rate of change of the square of the length of each link is zero. (More about this in Section 6.1 on page 51.) So, the requirement is that

$$(a_k(0) - a_l(0)) \cdot (a'_k(0) - a'_l(0)) = 0, \text{ for all connected } k, l, \tag{2.5}$$

as it was explained by Ben Roth in [29].

An immediate conclusion is that it is possible for a structure to be rigid but not infinitesimally rigid. However, no structure is infinitesimally rigid but flexible because a structure that admits finite motions certainly admits infinitesimal motions, as well.

A couple of examples of structures that are rigid but not infinitesimally rigid are given in Figure 2.2 on page 7. Both of those structures have infinitesimal flexibilities: the 2-D structure has a nontrivial velocity at a_2, while the 3-D pyramid has a nontrivial velocity at a_0. However, if the position of the node that allows nontrivial velocity is changed vertically by any nonzero amount, infinitesimal rigidity is achieved.

This is used in Chapter 8 on page 68 when a 3-D infinitesimally flexible structure is introduced that, because of the extra degree of freedom in a linearized setting, returns an extra compatibility condition. However, when its nodes are perturbed by any nonzero amount, the new structure becomes infinitesimally rigid, and returns the expected number of compatibility conditions. This is where another very useful definition of infinitesimal rigidity from linear algebra, given in Section 3.1 on page 9, is used.

Definition 5 (Rigid-Body Motion)**.** *$Q^d(S)$ is said to undergo a rigid-body motion if the distance between any two nodes stays the same, even though their position in space may change. In other words, $Q^d(S)$ preserves its shape, but may change its position.*

This holds for any d-dimensional body, as well. That is, if the distance between any two points in a body remains the same before and after the deformation, the body is said to have undergone a rigid-body motion.

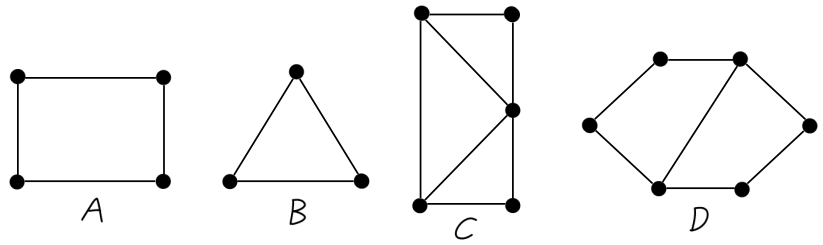

Figure 2.1: A and C are rigid, while B and D are flexible.

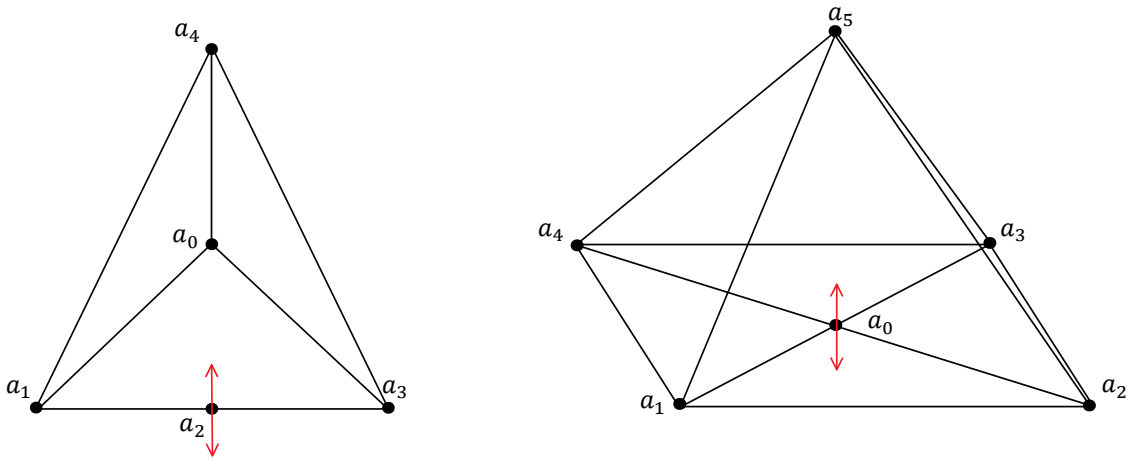

Figure 2.2: Both of these structures are rigid, but not infinitesimally rigid. The nontrivial velocities are denoted by the red arrows.

CHAPTER 3

COUNTING COMPATIBILITY CONDITIONS

James Clerk Maxwell in the mid-1800s was one of the first mathematicians to study the rigidity of structures. His rule for counting arguments, given in [23] and extended in [2] by C.R. Calladine, is used in this chapter, and they were, along with [17] and [27], very helpful in searching for ways to count the number of compatibility conditions of various structures by using their symmetry and geometry.

Gerard Laman, from the University of Amsterdam, studied the rigidity of graphs in the early 1970's. He built on Maxwell's work and coined the term Laman graphs [21], referring to minimally rigid 2-D graphs — and hence 2-D structures, using this document's definition — that have n nodes and exactly $2n-3$ links, which *is* the minimum number of links any generic structure with n nodes must have in order to be rigid. This fact is used throughout this chapter when attempting to count compatibility conditions of various structures, because adding new links to a minimally rigid structure is equivalent to adding compatibility conditions.

Donald Jacobs and Bruce Hendrickson [19] developed an algorithm — the pebble game — to check for rigidity of structures by counting the degrees of freedom, which was a significant improvement in the number of calculations from the Laman's theorem in 1972 that basically said that a minimally rigid structure, call it Q, is rigid if and only if for each substructure of Q with n' nodes and m' links it holds that $m' \leq 2n' - 3$.

In the pebble-game approach a minimally rigid structure is built from the bottom up, meaning one independent link after another is added until a minimally rigid structure is reached. In this document, however, the

approach is somewhat in reverse; namely, the number of compatibility conditions are found by removing links until a minimally rigid structure is reached.

3.1 Computing C_d Using Degrees of Freedom

In order to detect the flexes of structures, other than rigid-body motion, both translation and rotation need to be fixed using as few parameter-constraints as possible, if these flexes are to be useful in studying compatibility conditions. There are different ways of accomplishing this, depending on $Q^d(S)$, such as fixing one node completely, and the y-direction of another node, which would work for any $Q^2(S)$.

In a 2-D case, each node has two degrees of freedom, and at least three parameter constraints need to be imposed in order to stop the rigid-body motion. The number of nodes for any $Q^d(S)$ is n, by definition. Therefore, every $Q^2(S)$ starts with $2n$ degrees of freedom. After imposing the three parameter-restrictions, the number of degrees of freedom becomes $D_2 = 2n - 3$ [23].

In 3-D, each node has three degrees of freedom, and at least six parameter constraints need to be imposed in order to stop the rigid-body motion. Because every $Q^3(S)$ starts with $3n$ degrees of freedom, this number is reduced to $D_3 = 3n - 6$, after imposing six parameter restrictions [23].

In general, in order to prevent the rigid-body motion, $r \geq \frac{d(d+1)}{2}$, where r represents the number of fixed parameters. Therefore, a structure with a minimum number of parameter-constrains has exactly $D_d = dn - \frac{d(d+1)}{2}$ degrees of freedom.

Suppose $Q^d(S)$, with r fixed parameters preventing the rigid-body motion ($r \geq 3$ for $d = 2$ and $r \geq 6$ for $d = 3$), undergoes a flex. Then there are $dn - r$ degrees of freedom. Each link can be represented by an equation that relates displacements with elongations, which implies that there are m such equations; call them $g_1, \ldots, g_m$. This system of equations in matrix form becomes

$$A \cdot U = \Lambda, \tag{3.1}$$

where A is a matrix of links, consisting of node coordinates, U a matrix of unknown displacements, and Λ a matrix of link elongations.

Fixed parameters are either given as conditions on all or many displacements — for example, an equation that sets the total angular momentum to zero — or as fixed displacements — for example, $u_0 = 0$, where a_0 is fixed. In the former case, a new equation is added, while in the latter case, one unknown is deleted. But this is equivalent to adding a row to A or deleting a row of U, respectively. In either case, the number of rows of A minus the number of rows of U equals the number of links minus the number of degrees of freedom: $m - (dn - r)$. Therefore, A is $M \times N$, with $M - N = m - (dn - r)$, while U is $N \times 1$, and Λ is $M \times 1$ matrix of m link elongations, and $M - m$ rows of zeros. Then the following definition holds:

Definition 6. *A rigid structure with* $r \geq \dfrac{d(d+1)}{2}$ *fixed parameters is said to be* infinitesimally rigid *if* $\ker(A) = 0$, *or* A *has full rank. Otherwise, it is said to be* infinitesimally flexible.

For an infinitesimally rigid structure with m links and $r = \dfrac{d(d+1)}{2}$ fixed parameters it holds that $M = N$, and the number of links, all of which are linearly independent, is equal to the number of degrees of freedom: $m = dn - r$. The matrix of links of this structure, A, has full rank, $rank(A) = M$, and the kernel is trivial.

Removing one link from the structure is equivalent to deleting a row from A corresponding to the removed link, resulting in an underdetermined system; the rank is dropped by 1 and the kernel becomes nontrivial, implying the structure's flexibility.

On the other hand, adding a link is equivalent to adding a row to A, which results in an overdetermined system because the number of links, $m + 1$, is by 1 greater than the number of degrees of freedom. This newly added link, call it l_{m+1}, is a linear combination of a unique subset of the m linearly independent links. It cannot be written as a linear combination of

two different sets of links because of their linear independence.

Let $Q \subset E(m)$ be that set of links. Then we write $l_{m+1} = L(Q)$, or l_{m+1} is a linear combination of all the links from set Q. If a link that does not belong to set Q is removed, then a row corresponding to this link is deleted from A, making it a square matrix again. However, this time we have linear dependency because it still holds that $l_{m+1} = L(Q)$. Therefore, the rank drops by 1, the kernel becomes nontrivial, and the structure becomes flexible.

If, however, a link from Q is removed, l_{m+1} can no longer be written as a linear combination of these links, and, therefore, becomes itself linearly independent of all the other links. In addition, the row corresponding to this removed link is deleted from A, making it a square matrix of full rank again, thus rigid.

Therefore, the links from set Q are the possible removable links (any one link from this set can be removed without affecting the structure's rigidity), while those outside of it are the nonremovable links (removing any one link from this set makes a structure flexible).

Adding k links and no nodes to an infinitesimally rigid structure with the same number of links as degrees of freedom results in k linearly dependent equations. In fact, we get an overdetermined system, and solving this system produces $M - N = k$ relations on the known elongations.

More precisely, for a d-dimensional *infinitesimally rigid* structure with n nodes, m links, and r fixed parameters, the number of these relations is, by [23]

$$C_d = m + r - dn, \tag{3.2}$$

which is the same as the difference between the rows and columns of matrix A, namely $C_d = M - N$, because A has full rank. Therefore, $C_d = M - N$ represents the number of extra conditions or links that keep the structure rigid. These conditions are called *compatibility conditions*. Geometrically, they are the necessary and sufficient conditions for $Q^d(S)$ to stay in a compatible state under any deformation.

However, it may also be possible to make this strengthened structure flexible after a removal of only one link, other than one of k's, if this link is not involved in any of the compatibility conditions, thus it is not used in a linear combination of any of the added links. For example, the structure in Figure 3.1 on page 25 does not become flexible if $l_{0,1}$ and $l_{0,12}$ are removed, say, but it does if $l_{6,7}$ is removed. But every structure has a certain maximum number of links that can be removed and still keep its rigidity; and that is precisely what C_d is.

This work shows the following theorem, which can be summarized by equation 3.2:

Theorem 3.1 (C_d and the Loss of Links)**.** *The number of compatibility conditions for a given $Q^d(S)$, denoted by C_d, is equal to the* maximum *number of links that can be removed from Q^d, without making it flexible.*

Note that the maximum in this theorem implies that the gradual decrease of C_d depends greatly on which links are being removed, because it is possible to make a rigid structure with $C_d > 1$ flexible after removing only one link, as is demonstrated by Figure 3.1 on page 25

However, for any $Q^d(S)$ with C_d compatibility conditions, there always *exist* C_d links that, when removed, do not affect the rigidity of that $Q^d(S)$, which is precisely what Theorem 3.1 says. For example, the structure in Figure 3.1 on page 25 has two inner nodes, which translates into two compatibility conditions, as shown in Theorem 3.2 on page 15. Therefore, there exist two links, which may not be unique, that can be removed, yet the structure remains rigid. However, if *any* three links are removed, the structure becomes flexible.

On the other hand, C_d does not define the minimum number of redundant links. In the case of Figure 3.1 on page 25, the minimum number is zero, because the removal of $l_{6,7}$ would make this structure flexible.

Furthermore, it is not possible to jump from $C_d = x$ to $C_d = x - y \geq 0$, where $y > 1$, with a removal of just one link, even though a structure with

$C_d > 0$ can become flexible after a removal of only one link; or, equivalently, it cannot lose more than one compatibility condition. This is because removing one link is equivalent to reducing the number of rows of A by one, so $C_d = M - N - 1$. For example, the two rigid *wheels* of the structure in Figure 3.1 on page 25 both still have one compatibility condition each, because they still have one inner node each, even after the removal of the link which makes the whole structure flexible. In fact, the removal of *one* link either reduces C_d by one, or keeps C_d the same and *removes* rigidity of the structure.

The number of compatibility conditions depends on r, as well. Namely, if, in addition to fixing one node plus the total angular momentum, which is three parameters altogether, one more node is fixed, or two additional parameters, the number of compatibility conditions rises by two, and it means that two additional links can be removed without losing the structure's rigidity.

In principle, the removal of all the links of a structure with all of its nodes fixed does not affect its rigidity because all links are *dead*. This is why, in order to study the *global rigidity* [12] of structures, the minimum number of parameters is fixed; namely, $r = \frac{d(d+1)}{2}$. Fixing any additional parameters adds that many compatibility conditions, including the trivial ones represented by dead links, which are the links between two fixed nodes.

The conditional rigidity, on the other hand, depends on r, and on the loading. This is explored in Section 9.3 on page 76.

3.2 C_2 for Triangulated Structures

The rest of this document, except for a brief visit of complete graphs in Section 3.5 on page 23, deals with triangulated structures, which is why a careful approach to its definition is used.

Definition 7 (Simple Polygonal Structure)**.** *A simple polygonal structure, Q_P, is a closed region of the plane bounded by a finite collection of linked*

nodes. [15] (Figure 3.2 on page 25)

A *diagonal link* of any polygonal structure Q_P is a link between its two nodes such that it lies in the interior of Q_P. A diagonal link does not touch the boundary of Q_P, except for its endpoints. Two diagonal links are noncrossing if they share no interior points [15].

A polygonal structure A in Figure 3.3 on page 26 has a diagonal link, $l_{4,6}$, while the link connecting a_2 and a_7 in B is *not* a diagonal link because it does not lie in the interior of the polygonal structure. C shows two intersecting diagonals.

In order to triangulate a polygonal structure it needs to be decomposed into triangles. And indeed

Definition 8. *A triangulation of a polygonal structure is a decomposition of it into triangles by a maximal set of noncrossing diagonal links.*

In other words, no more noncrossing diagonal links can be added to the set. Triangulation of a polygonal structure is not unique, in general. Figure 3.4 on page 26 shows three different triangulations of the same Q_P.

It is a known fact that every polygonal structure has a triangulation; in fact, even every polygonal structure with holes, such as D in Figure 3.2 on page 25, admits triangulation. Furthermore, every triangulation of Q_P with n nodes has $n-2$ triangles (or faces) and $n-3$ diagonal links. (For proofs of these go to [15], pages 4 and 5.)

In general, however, the following definition holds:

Definition 9 (Triangulation)**.** *A triangulation of a set $S^2(n)$ with a polygonal boundary Q_P is a subdivision of the plane (bounded by Q_P) by a maximal set of noncrossing links whose set of nodes is $S^2(n)$, and its boundary is Q_P. [15]*

This ends up being a decomposition of $S^2(n)$, where its boundary nodes are restricted by Q_P, into triangles. To demonstrate what *maximal* in this definition means, look at Figure 3.5 on page 27.

Note that every node, except for a_0, is a boundary node, and contributes to the polygonal boundary, Q_P, whose boundary links are those connecting the boundary nodes. It is easy to see that A is not a triangulation because it is not composed of triangles — the quadrilateral $a_4a_5a_6a_7$ is not a triangle. Connecting a_5 to a_7 is not enough, because B is not yet a triangulation, even though it is composed of triangles. That is because B is not a *maximal subdivision.* Namely, one more link can be added which would not intersect any other link — $l_{3,7}$ — making C a triangulation. Triangulations of a given set $S^2(n)$ are generally not unique: C and D are both triangulations of the same set $S^2(9)$.

In an infinite, periodic, hexagonal lattice given in Figure 3.6 on page 27, each node has two degrees of freedom, and is connected to six other nodes. There is one link connecting two nodes, therefore there are three links for each node.

For a large sample, where the elements of periodicity have n nodes, there are $3n$ links, and $C_2 = 3n - 2n = n$ because the boundary conditions are ignored. This implies that the number of compatibility conditions of a triangular grid equals the number of inner nodes, which was formerly stated, and is proved next.

Let $Q^2_\tau(S)$ be a triangulated structure, with a polygonal boundary. Then the following theorem holds:

Theorem 3.2. *Let $n_i + n_b = n$, where n_i is the number of inner nodes, while n_b is the number of outer, or boundary nodes of a given $Q^2_\tau(S)$. Then*

$$C_2 = n_i. \tag{3.3}$$

Proof. Let $Q^2_\tau(S)$ be given, and let m_b be the number of boundary links, and m_i the number of inner links. Then $m = m_i + m_b$ and $m_b = n_b$. Because $Q^2_\tau(S)$ is a triangulation, each inner link *bounds* two faces (or triangles), and each boundary link bounds one face. Then the number of *bounded*

faces is given by $3F = 2m_i + m_b$. The Euler's Characteristic Formula[1] says that

$$2 = n - m + F$$

for all connected planar graphs, so it holds for $Q^2_\tau(S)$, as well. This, however, includes the unbounded face outside $Q^2_\tau(S)$. Therefore, for bounded faces (or triangles) it holds that

$$\begin{aligned}
1 &= n - m + F \\
&= n_i + n_b - m_i - m_b + \frac{2}{3}m_i + \frac{1}{3}m_b \\
&= n_i + n_b - m_i - m_b + \frac{2}{3}m_i + \frac{1}{3}n_b \\
&= n_i + \frac{1}{3}n_b - \frac{1}{3}m_i,
\end{aligned}$$

or $3 = 3n_i + n_b - m_i$. But according to Theorem 3.1 on page 12,

$$C_2 = m - 2n + r, \tag{3.4}$$

where r represents the number of fixed parameters. for a general $Q^2(S)$, $r = 3$, and thus

$$\begin{aligned}
C_2 &= m - 2n + 3 \\
&= m_i + m_b - 2n_i - 2n_b + 3 \\
&= m_i + n_b - 2n_b - 2n_i + 3 \\
&= -2n_i + m_i - n_b + 3 \\
&= -2n_i + m_i - n_b + 3n_i + n_b - m_i \\
&= n_i.
\end{aligned}$$

□

This theorem makes counting compatibility conditions fairly easy. For example, triangulations C and D in Figure 3.5 on page 27 have only one inner node, hence only one compatibility condition.

---

[1]The Euler's Characteristic Formula is shown using induction, and can be found in many applied math textbooks, one of which is [15], p. 63.

This is called the *global*, as opposed to conditional, number of compatibility conditions of a given structure, and it is equal to the number of compatibility conditions any structure has with the minimum number — which is three for 2-D and six for 3-D — of parameters fixed in order to prevent rigid body motion, and it indicates the maximum number of links that can be removed from the structure, while still keeping its global rigidity.

However, this theorem does not hold if more than three parameters are fixed, as was seen in the preceding section. In fact, each new fixed parameter adds another compatibility condition, where dead links count as trivial compatibility conditions, because each added parameter adds another equation.

In particular, Theorem 3.2 on page 15 holds for a structure in Figure 3.7 on page 28 which has two inner nodes, hence two compatibility conditions.

On the other hand, the structure in Figure 3.8 on page 28 has five inner nodes, hence five compatibility conditions.

3.3 Triangulations with Holes

Theorem 3.2 on page 15 only works for $Q^2_\tau(S)$ (triangulated structures with a polygonal boundary), and breaks down for triangulations with holes, which can be demonstrated by a structure shown in Figure 3.9 on page 29. It has no inner nodes between the inner and the outer boundaries; so, according to Theorem 3.2 on page 15, there should be no compatibility conditions, which is equivalent to saying that the removal of any link would make this structure flexible. But this is clearly not the case. In fact, one can calculate what C_2 of this structure is by "filling in" the hole with links.

In fact, it takes exactly five links to fill in the hole, and the structure becomes Q^2_τ (Figure 3.10 on page 29), which means that Theorem 3.2 on page 15 holds, implying that $C_2 = 8$ for this new structure, because there are eight inner nodes.

But adding five links and no nodes is equivalent to adding five com-

patibility conditions. Thus, the original structure, given in Figure 3.9 on page 29, has three compatibility conditions.

In fact, it turns out that a more general statement, regarding triangulated structures similar to that given in Figure 3.9 on page 29, holds. But first, a definition:

Definition 10 (Tire-Track)**.** *Any 2-D, closed, rigid structure, consisting of triangles, having two distinct sets of boundary nodes and links, the inner and the outer, and no inner nodes, is called a tire-track, and is denoted by* Q^2_Δ.

The structure given in Figure 3.9 on page 29 is an example of Q^2_Δ. Another, more general example is given in Figure 3.11 on page 30.

The simplest possible tire-track has only seven nodes, four inner-boundary nodes and three outer-boundary nodes (Figure 3.12 on page 30).

Lemma 3.3. *Every* Q^2_Δ *has exactly three compatibility conditions.*

Proof. This proof is by observation. Namely, one can see that removing any outer-boundary link from Q^2_Δ cuts the tire-track into one "connected sausage," which is rigid. Removing another outer-boundary link divides the tire-track into two connected sausages, both of which are rigid, making the entire structure rigid, as well. The removal of the third outer link, the structure is reduced to three connected sausages, each rigid. The whole structure is mechanically equivalent to a triangle, demonstrated by dotted lines in Figure 3.13 on page 31. If any other link is removed, the structure becomes mechanically equivalent to a square, which is flexible. This means that the structure in Figure 3.13 on page 31 must have $C_2 = 0$. It then follows from Theorem 3.1 on page 12 that $C_2 = 3$ for Q^2_Δ, or $C\left(Q^2_\Delta\right) = 3$. □

A different proof:

Proof. Let b_i be the number of inner-boundary nodes of any Q^2_Δ. Then $b_i \geq 4$, because there is no hole for $b_i = 3$. But any tire-track can be transformed

into a no-hole $Q^2_\tau(S)$ by simply filling in the hole with links, until a complete triangulation is reached. A tire-track with $b_i = 4$ needs to have only one link added in order to complete the transformation (Figure 3.14 on page 31). The transformed structure now has $C_2 = 4$, because it has $n_i = 4$. But adding *one* link adds exactly *one* compatibility condition, as long as no new nodes are added; so the original tire-track must have three compatibility conditions.

A $b_i = 5$ tire-track (Figure 3.15 on page 32) is constructed by adding a node between any two inner-boundary nodes, and connecting it with its neighbors. (Link $l_{1,6}$ is necessary in order to preserve triangulation.) In total, our new $b_i = 5$ tire-track has one additional node, or two new parameters, and two new links; so the rank, nullity, and the number of compatibility conditions stays the same in order to fill in $b_i = 5$ tire-track's hole we need to add only one more link than for $b_i = 4$ tire-track, or two in total, and it is the one that connects the two nodes in between which the fifth inner-boundary node is located, or $l_{0,2}$ in Figure 3.15 on page 32. The two additional links add two compatibility conditions to the $b_i = 5$ tire-track, and because $n_i = 5 \Rightarrow C_2 = 5$, the original tire-track must have $5 - 2 = 3$ compatibility conditions.

Adding inner-boundary nodes and links in this fashion leads, inductively, to the fact that, in order to transform any $Q^2_\Delta(S)$ with b_i inner-boundary nodes into a no-hole $Q^2_\tau(S)$, exactly $b_i - 3$ links needs to be added,[2] which is equivalent to adding $b_i - 3$ compatibility conditions. But now this no-hole $Q^2_\tau(S)$ has b_i inner nodes, which implies that it now has b_i compatibility conditions. Therefore, the original b_i tire-track has $b_i - (b_i - 3) = 3$ compatibility conditions. □

Aside from these geometrically observable proofs, an easy check is by applying equation 3.2 again:

---

[2]This does not include the two links added every time a new node is added in order to connect it with its neighbors.

First notice that, for a tire-track with b_i inner-boundary nodes and b_o outer-boundary nodes,

$$m_b = m_i + m_o = b_i + b_o = n,$$

or the total number of boundary links is equal to the inner-boundary links + the outer-boundary links, which happens to be n, or the total number of nodes.

Additionally, there are

$$2b_i - (b_i - b_o) = b_i + b_o = n \tag{3.5}$$

links that connect b_i's and b_o's that are necessary for maintaining a triangulation. (This is seen by zig-zagging from b_i's onto b_o's, and then adding — that may turn into subtracting if $b_o < b_i$ — those b_o's that are left.) Therefore, $m = 2(b_i + b_o) = 2n$. For a globally rigid tire-track, $r = 3$. Therefore,

$$\begin{aligned} C_2 &= m - 2n + r \\ &= 2n - 2n + 3 = 3. \end{aligned}$$

Combining Theorem 3.2 on page 15 and Lemma 3.3 on page 18 gives us the following

Theorem 3.4 (C_2 for Structures with a Hole)**.** *The number of compatibility conditions of any triangulated 2-D structure with a single "hole," or a structure which has distinct inner- and outer-boundary nodes and links, is equal to the number of inner nodes plus 3. That is* $C_2 = n_i + 3$.

Here structures are not limited to tire-tracks. In fact, Theorem 3.4 says that any structure with distinct inner- and outer-boundary nodes, no matter how thick it is, must have $C_2 = n_i + 3$. This is true for tire-tracks as well, because they have $n_i = 0$, and we are back to Lemma 3.3 on page 18.

Proof. Suppose a structure has n_i inner nodes, b_o outer-boundary nodes, and b_i inner-boundary nodes. Then the claim is that $C_2 = n_i + 3$. The same procedure that is used in the second proof of Lemma 3.3 on page 18

implies the result. Namely, $b_i - 3$ links need to be added until a no-hole $Q^2_\tau(S)$ is reached, which translates into $b_i - 3$ new compatibility conditions. Because there are now a total of $n_i + b_i$ inner nodes, and a total of $b_i - 3$ new compatibility conditions, the original structure has

$$C_2 = (n_i + b_i) - (b_i - 3) = n_i + 3$$

compatibility conditions. □

3.4 C_d for Complete Graphs

A complete graph is a maximally connected Q^d, or a structure where every node is connected to every other node. In this case, it is quite possible to have many intersecting or overlapping links. Like, for example, in Figure 3.16 on page 32.

In the figure, nodes a_3 and a_6 are connected by a link, as well as nodes a_2 and a_5, and nodes a_1 and a_4, and these three links are overlapped by the pairs of links connecting these three pairs of nodes via a_0. So, for example, there are 3 links between a_3 and a_6: one link connecting a_3 and a_0, another link connecting a_0 and a_6, and a third link connecting a_3 and a_6.

Another example of a complete graph is given in Figure 3.17 on page 33. This structure has eight nodes, and there are no overlapping links.

Complete graphs can have many more links, thus compatibility conditions, than triangulated structures; but for complete graphs with $n \leq 4$ the number of links is equal to or greater than the number of links of triangulated structures with the same number of nodes. In all other cases the number of links of a complete graph is strictly greater than the number of links of triangulated structures with the same number of nodes. The structure in Figure 3.18 on page 33 is an example of a structure that is both a complete graph and a triangulation.

Because each node is connected to every other node, and there are n different nodes in any given graph, there are $n - 1$ connections for each node; but because $L_{i,j} = L_{j,i}$, the number of links is

$$m = \frac{n(n-1)}{2}. \tag{3.6}$$

But how many compatibility conditions does a complete graph have? First note that $C_2 = 0$ if $n < 4$, and $C_3 = 0$ if $n < 5$, because then no inner nodes are possible, and every link breakage would make Q^d, $d = 2, 3$, flexible.

Equation 3.2, namely the fact that $C_2 = m - 2n + 3$ and $C_3 = m - 3n + 6$, implies that

$$\begin{aligned} C_2 &= m - 2n + 3 \\ &= \frac{n(n-1)}{2} - 2n + 3 \\ &= \frac{(n-2)(n-3)}{2} \end{aligned} \tag{3.7}$$

and

$$\begin{aligned} C_3 &= m - 3n + 6 \\ &= \frac{n(n-1)}{2} - 3n + 6 \\ &= \frac{(n-3)(n-4)}{2}. \end{aligned} \tag{3.8}$$

But then (3.7) implies that

$$\begin{aligned} C_2 &= m - 2n + 3 \\ &= \frac{n(n-1)}{2} - 2n + 3 \\ &= \frac{(7-2)(7-3)}{2} = 10 \end{aligned}$$

for the structure in Figure 3.16 on page 32. This can be checked geometrically, as well, by removing the extra nine links added to a hexagon, which is known to have one compatibility condition, because it has only one inner node.

Continuing in the fashion of the last chapter, if a triangulation is made from the structure in Figure 3.17 on page 33 by removing all the extra links, a structure in Figure 3.19 on page 34 is obtained, which has zero compatibility conditions because it has no inner nodes. Because fifteen

links are removed, and a removal of one link is equivalent to removing one compatibility condition, the conclusion is that the structure in Figure 3.17 on page 33 has fifteen compatibility conditions. And, indeed, using (3.7) again

$$\begin{aligned} C_2 &= m - 2n + 3 \\ &= \frac{n(n-1)}{2} - 2n + 3 \\ &= \frac{(8-2)(8-3)}{2} = 15, \end{aligned}$$

yields the same result.

3.5 Compatibility Conditions Near the Fixed Nodes

Assume that a triangular grid, like the one shown in Figure 3.20 on page 34, is being fixed along one of its edges, say all the first-column nodes are fixed in both directions. How does one count compatibility conditions along this edge? What roles do the links attached to the fixed nodes on one end, and the free node on the other, play in determining the number of compatibility conditions?

Let us look at the basic structure depicting this situation: two fixed nodes and one free node connected in a triangle. This structure has no compatibility conditions, but it is, of course, rigid. However, adding another such structure next to it would add three free links, and one free node, resulting in one compatibility condition. By repeating this process, one compatibility condition per structure is added; so, in the end, the number of compatibility conditions is equal to the number of free nodes minus one.

This can be demonstrated by the structure in Figure 3.21 on page 35. In fact, if we assume that only $\Delta a_1 a_5 a_6$ is given, which represents our first basic structure, where a_1 and a_5 are fixed, while a_6 is free, and ignore the rest of that structure, then one fixed node, a_9, is added, and one free node, a_{10}, is added, connected to $\Delta a_1 a_5 a_6$ via three free links, l_{12}, l_{16}, and l_{21}. It is important to notice that the fixed links and nodes do not contribute to

compatibility conditions, because they would not break or move under any loading.

This new structure, $a_1a_6a_5a_{10}a_9$, is rigid plus one; that is, it has one compatibility condition. Adding the other two free nodes, a_{14} and a_{18}, and the two fixed nodes, a_{13} and a_{17}, together with six free links, l_{22}, l_{26}, l_{31}, l_{32}, l_{36}, l_{41}, and two more fixed links, l_{23} and l_{33}, completes the structure, and adds two more compatibility conditions: six free links minus two free nodes, which have four degrees of freedom, equals two compatibility conditions.

Interestingly enough, adding the free node a_2 to our structure does not change the number of compatibility conditions because of the two free links, l_1 and l_2, which cancel with the two degrees of freedom of a_2, so the whole structure consisting of the first two columns in Figure 3.21 on page 35, where the first column is fixed while the second one is free, has exactly 3 compatibility conditions.

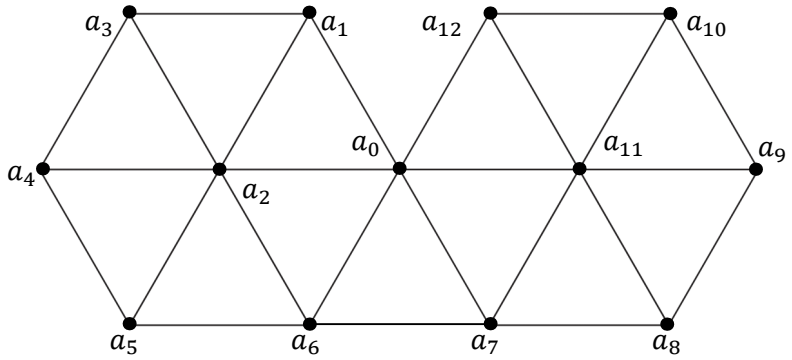

Figure 3.1: This structure has two inner nodes, a_2 and a_{11}, hence 2 compatibility conditions. (This is a theorem that is proved in the next section.) However, it loses its rigidity after $l_{6,7}$ is removed.

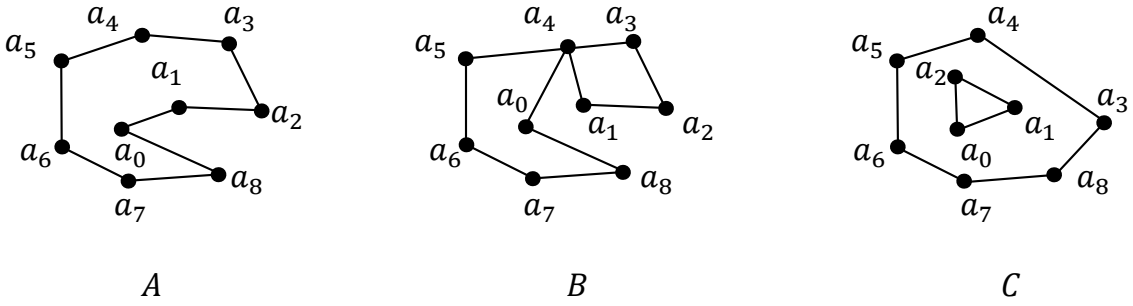

Figure 3.2: Three different lattices; only A is polygonal.

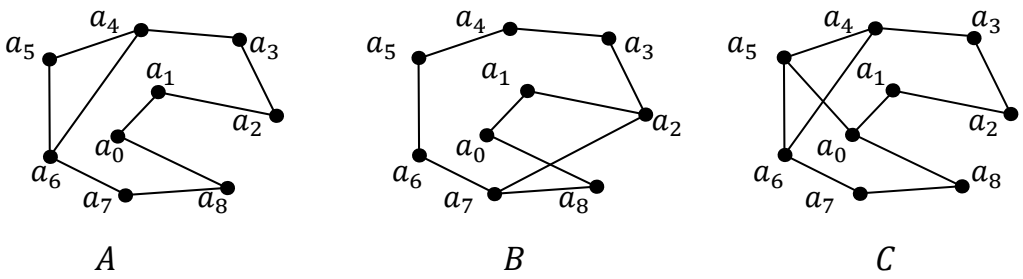

Figure 3.3: A polygonal structure with the diagonal link, A, not a diagonal link, B, and to crossing data links, C.

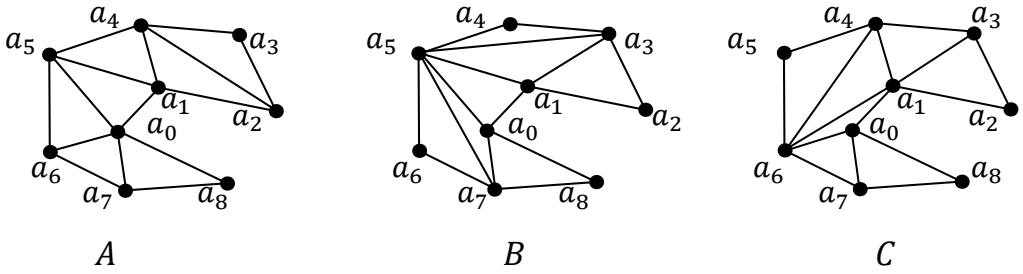

Figure 3.4: Three different triangulations of the same Q_P.

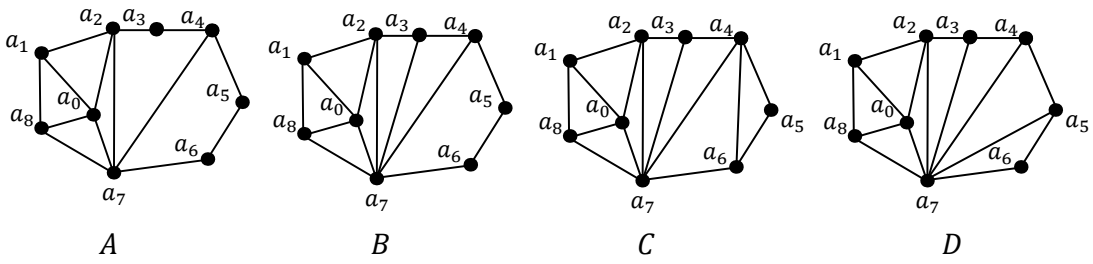

Figure 3.5: A and B are not triangulations, while C and D are.

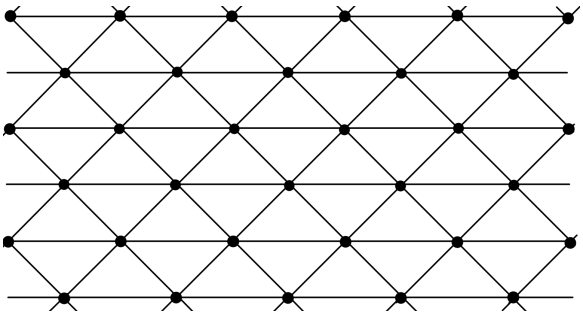

Figure 3.6: This is an infinite periodic triangular grid.

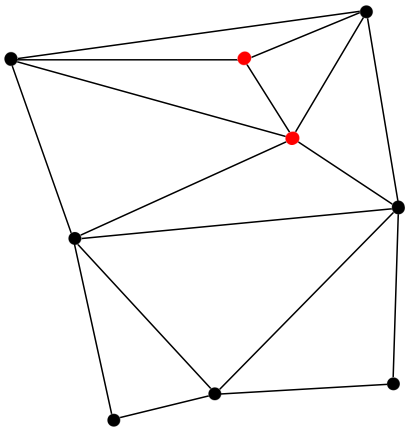

Figure 3.7: This triangular structure has two inner nodes, hence two compatibility condition.

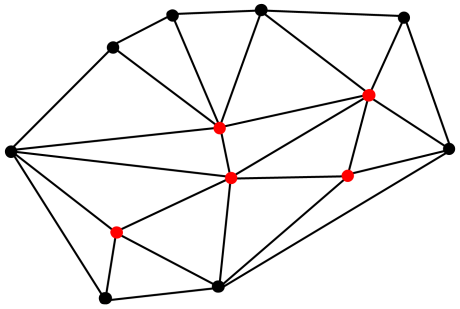

Figure 3.8: The red nodes are the inner nodes; therefore, this structure has five compatibility conditions. In other words, at most five links can be removed for it to stay rigid, while removing any sixth link would make it flexible.

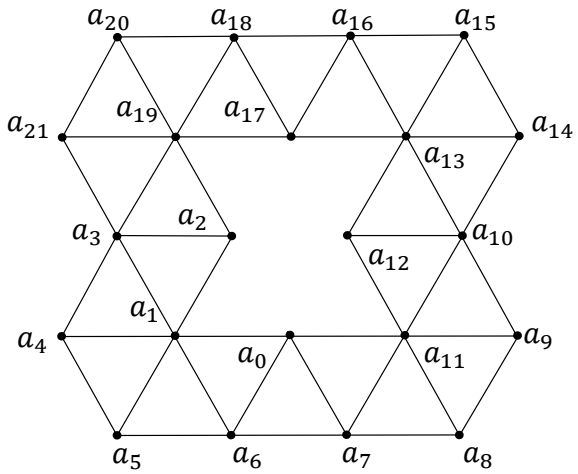

Figure 3.9: Polygonal triangulated structure with a hole. It has zero inner nodes.

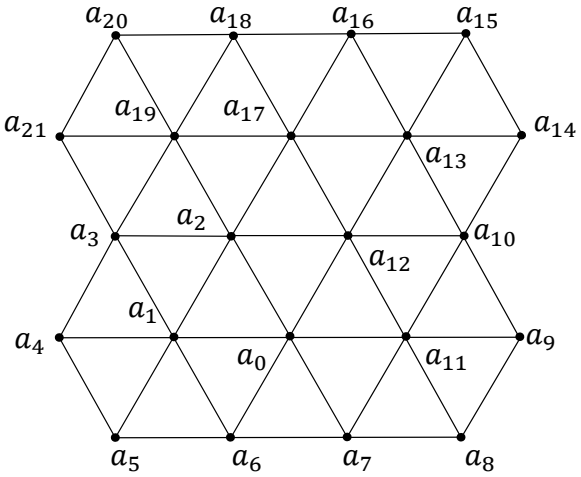

Figure 3.10: This structure has 8 inner nodes, which implies that $C_2 = 8$.

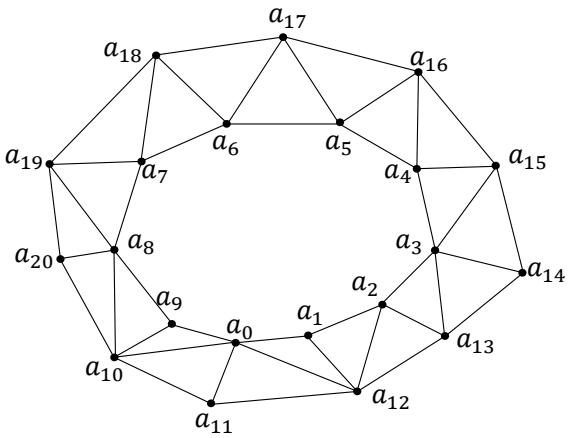

Figure 3.11: This is a tire-track. Notice the two distinct boundary nodes and links: the inner-boundary nodes are a_0 to a_9, while the outer-boundary nodes are a_{10} to a_{20}.

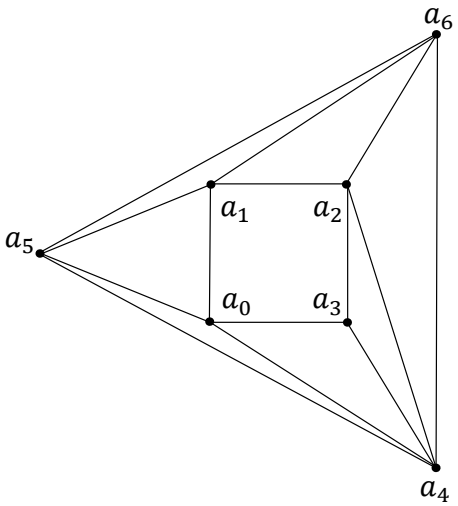

Figure 3.12: This is the simplest tire-track, as far as the number of nodes and links are concerned.

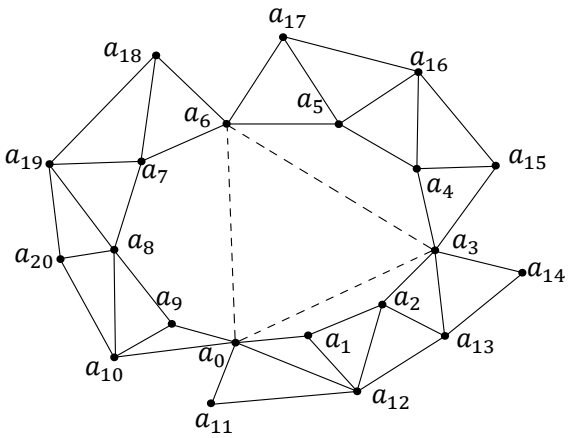

Figure 3.13: Three connected sausages. It is rigid because it is mechanically equivalent to a triangle.

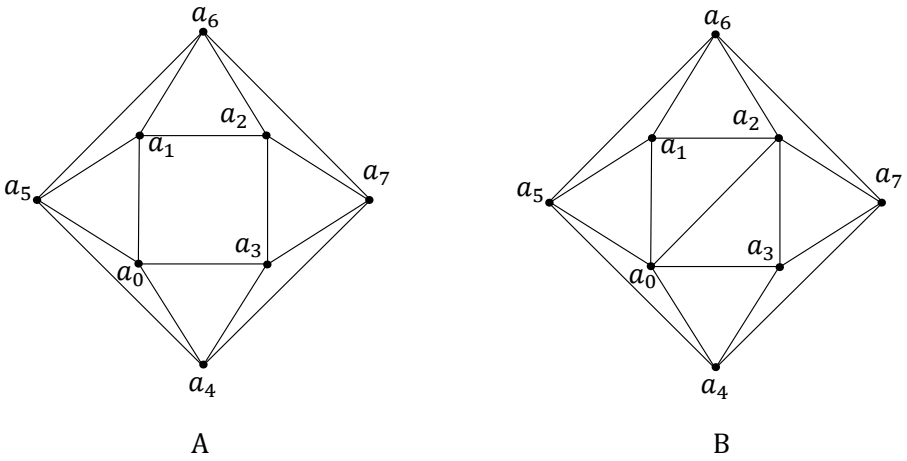

Figure 3.14: A is a tire-track, or Q^2_Δ, while B is a triangulated structure without a hole (no-hole $Q^2_\tau(S)$), which has 4 inner nodes, hence 4 compatibility conditions, and it is obtained by adding $l_{0,2}$ to structure A.

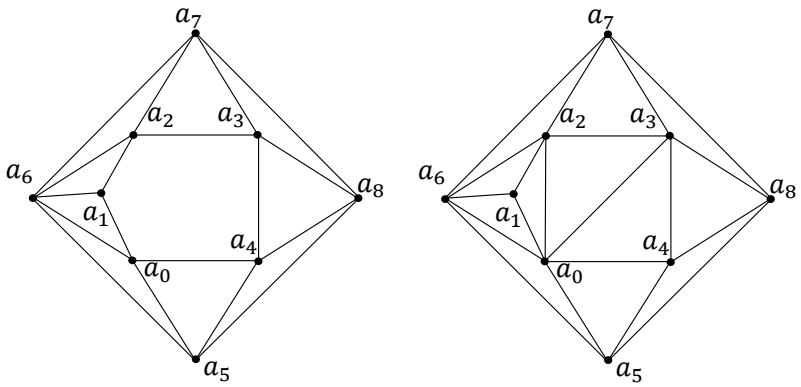

Figure 3.15: Tire-track with $b_i = 5$ transformed into a no-hole $Q^2_\tau(S)$.

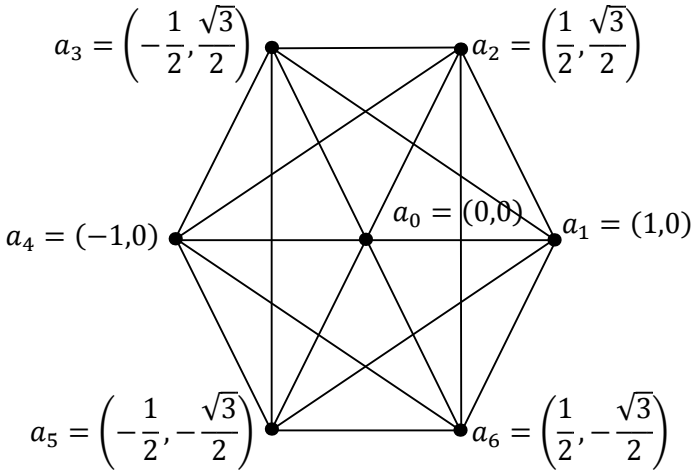

Figure 3.16: This is a complete graph, where every node is connected to every other node. Note, for example, that nodes a_3 and a_6 are connected by a link, as well as nodes a_2 and a_5, and nodes a_1 and a_4, and these three links are "hidden" from the view by the pairs of links connected through a_0. In total there are 21 links.

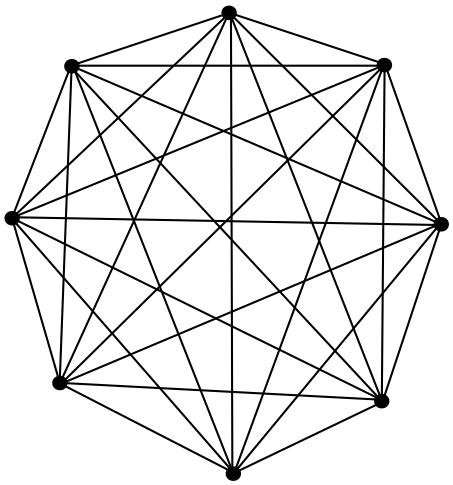

Figure 3.17: A Complete Graph with Eight Nodes.

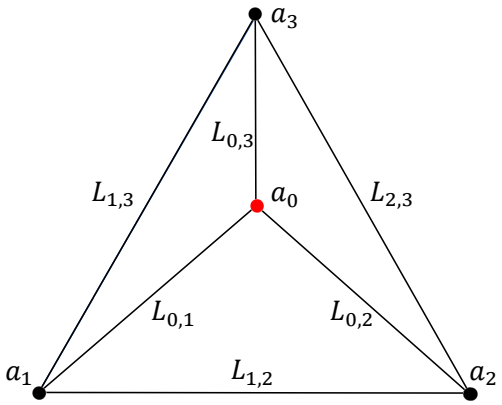

Figure 3.18: This is the simplest 2-D structure with $C_2 > 0$. It is also the only generic structure that is both a complete graph and a triangulation.

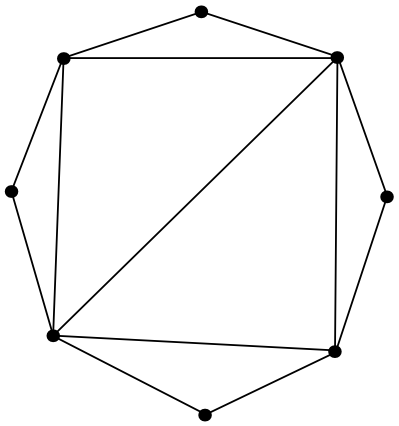

Figure 3.19: A triangular structure having no compatibility conditions.

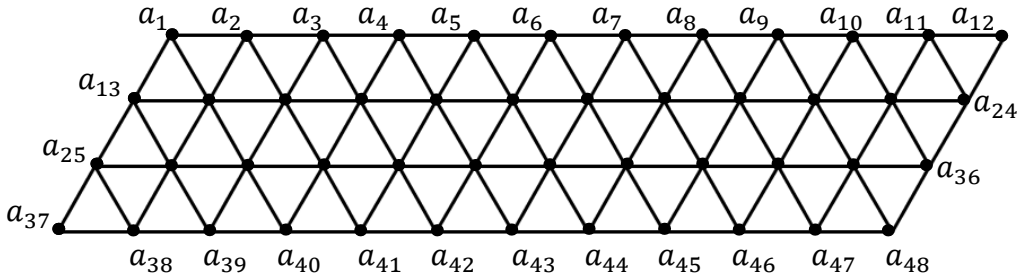

Figure 3.20: This is a 4×12 triangular grid, which is denoted by $Q_\tau^2(S_{4\times 12})$.

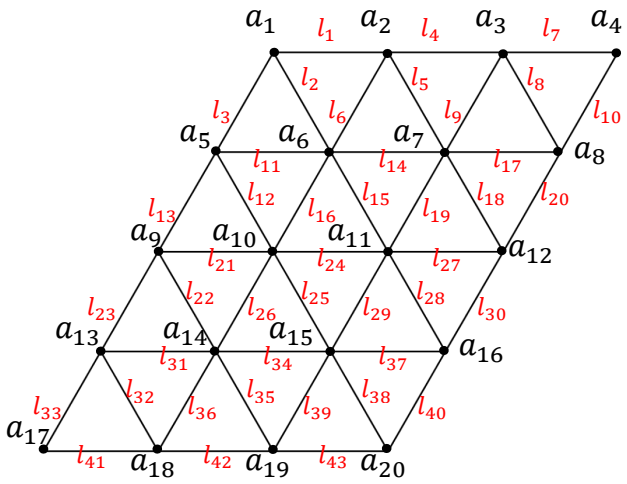

Figure 3.21: Numbering of the links of $Q^2_\tau(S_{5\times 4})$.

CHAPTER 4

FINDING COMPATIBILITY CONDITIONS

So far the only discussion was about various ways of computing the number of global discrete compatibility conditions, not how to represent them. Namely, discrete compatibility conditions are given in the form of equations, and they have geometrical meaning.

Triangulations have as many compatibility conditions as they have inner nodes. In this and other chapters the main focus is on finding the one compatibility condition of triangulations with one inner node.

Among the most helpful materials in computing and understanding discrete compatibility conditions proved to be M. Braun's article [1], as well as the PhD thesis by András Lengyel [22]. The classic works by J. Clerk Maxwell [23] and M.F. Thorpe [32] are used throughout this chapter as well as this entire document.

4.1 Nonlinear Discrete Compatibility Equations

Recall that a compatibility condition simply means a condition that has to be satisfied if a structure is to stay in a compatible state. For discrete structures this means that, when nodes are displaced, the links follow them; in other words, there are no detached links or overlapping nodes. Keeping this in mind, the following equation has to hold for each inner node of any triangulation

$$\cos\left(\sum_{i=1}^{k}\alpha_i(j)\right)=1 \qquad j=1,\ldots,n_i \tag{4.1}$$

where k is the number of angles that k links connected to the inner node j make. This condition simply says that the sum of all angles between all the links connecting any inner node is always 2π. For example, the most

simple triangulation with one inner node has three links connected to it, and has, therefore, three angles around it (Figure 4.1 on page 41).

The next observation is that each of these angles can be expressed in terms of the lengths of links using the law of cosines. If a structure is to stay in a compatible state, the links have to follow the law of cosines even after the deformation.

In order to compute a compatibility condition of a triangulated k-gon with one inner node, one first needs to write equation (4.1) as

$$\cos\left(\sum_{i=1}^{k}\alpha_i\right) = 1, \tag{4.2}$$

because this structure has only one inner node and k links connected to it, opening just as many angles. Then, it follows by the law of cosines that

$$\cos\alpha_j = \frac{L_{0,j}^2 + L_{0,j+1}^2 - L_{j,j+1}^2}{2L_{0,j}\cdot L_{0,j+1}}, \tag{4.3}$$

for all $j = 1,\ldots,k$, where $j = ((j-1) \bmod k) + 1$. One has to be careful, however, when labeling the nodes and the links if one is to use equation (4.3). Namely, the inner node has to be labeled a_0, and the outer nodes have to be labeled consecutively, as shown in Figure 4.1 on page 41.

This means that the compatibility condition of this particular k-gon can be given in terms of its link lengths, because

$$\begin{aligned}\cos(\alpha+\beta) &= \cos\alpha\cos\beta - \sin\alpha\sin\beta \\ &= \cos\alpha\cos\beta - \sqrt{(1-\cos^2\alpha)\,(1-\cos^2\beta)},\end{aligned} \tag{4.4}$$

for any two angles α and β. If α represents the first $k-1$ angles, and $\beta = \alpha_k$, then by continuing this procedure, and by using the fact that $\sin\alpha_j = \sqrt{1-\cos\alpha_j}$, for any j, the following result is achieved: cosines of each individual angle on the right-hand side, and $\cos(\alpha_1+\cdots+\alpha_k)$ on the left-hand side, which is equal to 1. Therefore, equation (4.2) is given in terms of the link lengths of the structure, and represents the nonlinear 2-D discrete compatibility condition.

Interestingly enough, if radicals are excluded by using the symmetric polynomial technique, this equation can be written as a polynomial in $L_{i,j}$. Therefore, any nonlinear discrete compatibility condition can be written in the form of a polynomial in $L_{i,j}$.

4.2 Linear Discrete Compatibility Equations

If a structure undergoes infinitesimal deformation, then only the linear terms in the compatibility-condition polynomial matter. It is not neccessary to compute the whole compatibility polynomial in order to find the linear terms; there is a much simpler way that is demonstrated in the next few examples.

The law of cosines for triangulations with small deformation implies that

$$\cos\alpha_j = \frac{\left(L_{0,j}+\epsilon\lambda_j\right)^2 + \left(L_{0,j+1}+\epsilon\lambda_{j+1}\right)^2 - \left(L_{j,j+1}+\epsilon\mu_j\right)^2}{2\left(L_{0,j}+\epsilon\lambda_j\right)\left(L_{0,j+1}+\epsilon\lambda_{j+1}\right)} \tag{4.5}$$

for all $j = 1, \ldots, k$, where $j = ((j-1) \bmod k) + 1$, and $\epsilon\lambda_j$ and $\epsilon\mu_j$ represent the elongations of radial and circumferential links, respectively.

The simplest 2-D structure with just one compatibility condition is given in Figure 4.1 on page 41. This condition is still going to be equation (4.2), or simply

$$\sum_{j=1}^{3} \alpha_j = 2\pi. \tag{4.6}$$

But each of these angles can be written in terms of the lengths and elongations of the structure's links,

$$\alpha_j = \arccos\left(\frac{\left(L_{0,j}+\epsilon\lambda_j\right)^2 + \left(L_{0,j+1}+\epsilon\lambda_{j+1}\right)^2 - \left(L_{j,j+1}+\epsilon\mu_j\right)^2}{2\left(L_{0,j}+\epsilon\lambda_j\right)\left(L_{0,j+1}+\epsilon\lambda_{j+1}\right)}\right), \tag{4.7}$$

for all $j = 1, 2, 3$, and where $j = ((j-1) \bmod 3) + 1$.

Now, linearized deformation implies that the linear terms of the Taylor polynomial about $\epsilon = 0$ of the arccosines represent the angles in the linear

setting. And, if all the angles are added up, the compatibility condition is obtained:

$$\sum_{j=1}^{3}\left(\sqrt{3}\lambda_j - \mu_j\right) = 0, \text{ or}$$

$$\sqrt{3}\sum_{j=1}^{3}\lambda_j = \sum_{j=1}^{3}\mu_j. \tag{4.8}$$

That is, the weighted sum of the radial-link elongations is equal to the weighted sum of the circumferential-link elongations.

The same procedure can be done for any other triangulated polygon with only one inner node, like a square with diagonals (Figure 4.2 on page 41), for which the compatibility condition is found to be the following, using the Maple code in Appendix A on page 110:

$$\sum_{j=1}^{4}\left(\lambda_j - \frac{\sqrt{2}}{2}\mu_j\right) = 0. \tag{4.9}$$

The weights are interesting in both of these cases in that the elongations of the radial links are weighted with the undeformed lengths of the circumferential links while the elongations of the circumferential links are weighted with the undeformed links of the radial links. In general, this rule, which can be written as

$$\sum_{j=1}^{n}\left(L_{j,j+1}\cdot\lambda_j - L_{0,j}\cdot\mu_j\right) = 0, \tag{4.10}$$

where $j = ((j-1) \bmod n) + 1$, does not hold. In fact, the weights seem to depend on the geometry of a structure. For example, the situation is different for an octagon. Namely, the compatibility condition of an octagon with one inner node is computed to be

$$\sum_{j=1}^{8}\left(\left(2-\sqrt{2}\right)\cdot\lambda_j - L_{j,j+1}\cdot\mu_j\right) = 0, \;\; j = j+8. \tag{4.11}$$

In this case, the weights of the radial-link elongations are neither the undeformed lengths of the circumferential links, nor the undeformed lengths

of radial links. Surprisingly enough, the weights of circumferential-link elongations are the undeformed lengths of circumferential links. This just shows that there is a more complicated rule governing the weights than one might have thought.

On the other hand, a regular hexagon, whose links are the same length, has the following as its compatibility condition

$$\sum_{j=1}^{6} (\lambda_j - \mu_j) = 0. \tag{4.12}$$

This is the difference of weighted sums; but because every link is of the same length, and without loss of generality this length is prescribed to be equal to one, the weight on each link is equal to one. The hexagon is done in much more detail in Section 6.2 on page 53, where this particular discrete compatibility condition is named the *wagon-wheel condition*.

The same technique can be performed on triangulations with more than one inner node, because all that is needed are the prescribed lengths of a structure, and the fact that the angles of the links around each inner node sum up to 2π. Then the law of cosines is used to write each angle in terms of these lengths and their infinitesimal deformations, and the fact that, if a structure is to stay in a compatible state after the deformation, it has to satisfy certain conditions — the compatibility conditions.

However, this technique becomes increasingly complicated when used for nontriangulated structures, or for 3-D objects, because the angles in 3-D are not scalars, they are vectors. For computing compatibility conditions of these, and many other general structures, the technique from Chapter 6 on page 51 is used.

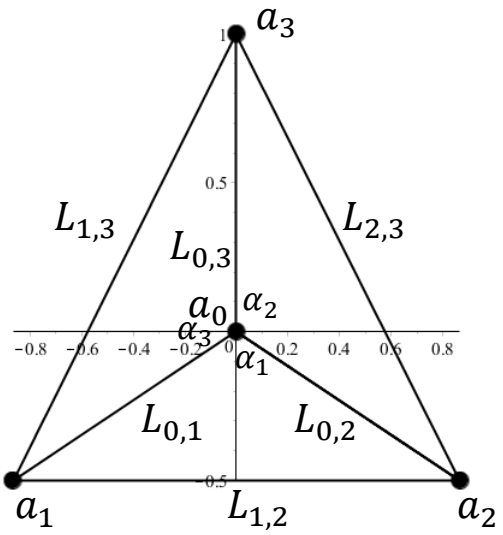

Figure 4.1: This structure has three links connected to the one inner node, and thus has three angles around it.

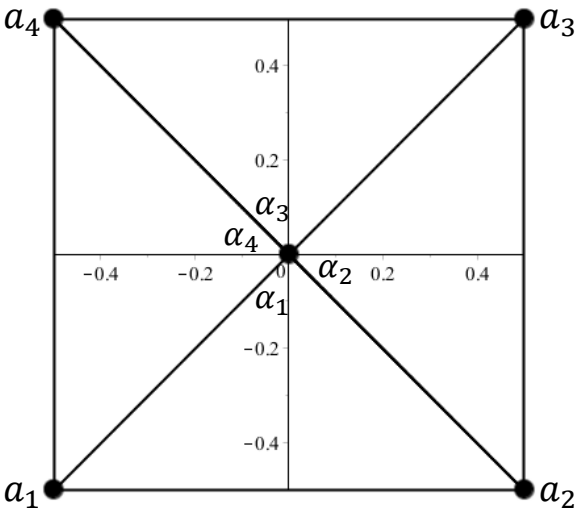

Figure 4.2: This square with one inner node and four inner links has only one compatibility condition.

CHAPTER 5

CONTINUOUS COMPATIBILITY CONDITIONS

The necessary conditions for continuum compatibility equations are derived in this chapter. The literature that is heavily used here includes the work by Martin Howard Sadd [9], and that by Pei Chi Chou and Nicholas J. Pagano [30]. The lecture notes by P. Kelly [20] proved to be helpful as well.

The last part of the chapter deals with an alternative approach to computing compatibility conditions using the Fourier transform. The equivalence of the Fourier transform approach to the traditional differentiation approach is shown in both 2-D and 3-D cases. In addition, the three fourth-order (in differentiation) independent continuum compatibility conditions in 3-D are derived from the six second-order conditions.

5.1 Strain-Displacement Relations

Elastic solids deform, i.e., the relative displacements between points in the body are changed, when external loadings are applied, in contrast to the rigid-body motion where the distance between points remains the same [30].

For nonlinear, or finite deformation, the difference between undeformed and deformed configurations can be very different. However, for linear deformation the difference is insignificant, and can be ignored.

Keeping this fact in mind, let P_0 and P be two neighboring points in a 3-D elastic body, and let P_0' and P' be the respected points in the same 3-D body after it underwent a linear deformation. Let r and r' be the respected relative position vectors, and u and u^0 the displacement vectors of P and P_0, respectively. Using the Taylor series expansion around P_0 implies

$$
\begin{aligned}
u_1 &= u_1^0 + u_{1,1}r_1 + u_{1,2}r_2 + u_{1,3}r_3 \\
u_2 &= u_2^0 + u_{2,1}r_1 + u_{2,2}r_2 + u_{2,3}r_3 \\
u_3 &= u_3^0 + u_{3,1}r_1 + u_{3,2}r_2 + u_{3,3}r_3,
\end{aligned} \tag{5.1}
$$

where the higher order terms of the Taylor expansion have been dropped due to linearity, making these components of r very small [30].

Now, combining the change in r,

$$\Delta r = r' - r = u - u^0$$

and 5.1 gives us the following equation

$$\Delta r = u_{i,j} r_j,$$

where $u_{i,j}$ is the displacement gradient tensor, and can be written as

$$
u_{i,j} = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} \\ u_{2,1} & u_{2,2} & u_{2,3} \\ u_{3,1} & u_{3,2} & u_{3,3} \end{bmatrix} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{bmatrix}, \tag{5.2}
$$

and can be decomposed into symmetric and antisymmetric parts

$$u_{i,j} = \varepsilon_{ij} + \omega_{ij}, \tag{5.3}$$

where

$$
\begin{aligned}
\varepsilon_{ij} &= \frac{1}{2}\left(u_{i,j} + u_{j,i}\right) \\
\omega_{ij} &= \frac{1}{2}\left(u_{i,j} - u_{j,i}\right),
\end{aligned} \tag{5.4}
$$

and where ε_{ij} is called the strain tensor, while ω_{ij} is called the rotation tensor [30]. The two can be written in a more familiar matrix notation form as

$$
\begin{aligned}
\varepsilon &= \frac{1}{2}\left(\nabla u + (\nabla u)^T\right) \\
\omega &= \frac{1}{2}\left(\nabla u - (\nabla u)^T\right).
\end{aligned} \tag{5.5}
$$

Looking at these strain-displacement relations more closely,

$$\begin{aligned}
&\varepsilon_{11} = u_{1,1}, \;\; \varepsilon_{22} = u_{2,2}, \;\; \varepsilon_{33} = u_{3,3} \\
&\varepsilon_{12} = \frac{1}{2}\left(u_{1,2} + u_{2,1}\right), \;\; \varepsilon_{23} = \frac{1}{2}\left(u_{2,3} + u_{3,2}\right), \;\; \varepsilon_{31} = \frac{1}{2}\left(u_{3,1} + u_{1,3}\right),
\end{aligned} \tag{5.6}$$

one concludes that there are six equations but only three displacement components (in 2-D there are three equations and two displacement components). This implies that the strains are not independent, but are related in some way, and these relations are the continuous compatibility conditions. Only if these conditions are satisfied can one be certain of continuous single-valued displacements.

5.2 Derivation of Continuous Compatibility Conditions

It was briefly mentioned in the first chapter what it means, geometrically, for a structure to stay in a compatible state after a deformation: in short, no gaps between neighboring point-masses of a continuum. This was also very nicely illustrated in section 2.6 of [30], which is what I am going to use in this paper. Namely, in Figure 5.1 on page 50, where a 2-D example is presented, there are two different deformations, continuous and single-valued in (c), and discontinuous in (d). The former case presents a continuous displacement field, where the elements are strained, taking into consideration their neighbors, while the latter case produces the discontinuous displacement field, with gaps between its neighboring elements.

In order to find the necessary conditions one needs to assume that the displacement field, u, is continuous and single-valued. Because the infinitesimal rotation tensor, ω, is given by

$$\omega = \frac{1}{2}\left(\nabla u - (\nabla u)^T\right),$$

then $\nabla u = \varepsilon + \omega$. Therefore,

$$\begin{aligned}
\nabla\omega &= \omega_{ij,k} \\
&= \frac{1}{2}\nabla_k\left(u_{i,j} - u_{j,i}\right) = \frac{1}{2}\left(u_{i,jk} - u_{j,ik}\right) \\
&= \frac{1}{2}\left(u_{i,jk} + u_{k,ij} - \left(u_{j,ik} + u_{k,ij}\right)\right) \\
&= \frac{1}{2}\left(u_{i,kj} + u_{k,ij} - \left(u_{j,ki} + u_{k,ji}\right)\right) \\
&= \varepsilon_{ik,j} - \varepsilon_{jk,i}, \qquad (5.7)
\end{aligned}$$

where continuous differentiability of ω is used. For the same reason, it follows that $\omega_{ij,kl} = \omega_{ij,lk}$. Therefore,

$$\omega_{ij,kl} - \omega_{ij,lk} = \varepsilon_{ik,jl} - \varepsilon_{jk,il} - \varepsilon_{il,jk} + \varepsilon_{jl,ik} = 0, \qquad (5.8)$$

which *are* the continuous compatibility condition equations in 2-D and 3-D. In tensor notation (5.8) becomes

$$\nabla \times (\nabla \times \varepsilon) = 0. \qquad (5.9)$$

The geometrical significance of compatibility conditions for a 3-D body can be seen by the following example, which was given by Pei Chi Chou and Nicholas J. Pagano in [9].

Let a 3-D body be divided up into small cubes, and let each cube be deformed by arbitrarily prescribed strained functions. Piecing up the deformed elements together, in an attempt to restore the strained body, leaves, in general, gaps and overlaps between the deformed cubes, just like there are gaps in between quadrilaterals in Figure 5.1 on page 50 for a 2-D body. They would only fit perfectly together if the strain components satisfy the compatibility conditions. However, if the displacement components are already single valued and continuous, then the strain components automatically satisfy the compatibility conditions.

Moving quickly away from the continuous setting for a moment, the geometrical significance of compatibility conditions for a structure is very similar: they represent those conditions under which the structure's nodes and links do not overlap or detach.

In order to show that these are also sufficient conditions, one needs to assume that (5.9) holds in some part of an elastic solid. Then the claim is that this condition is sufficient to guarantee the existence of a continuous, single-valued displacement field, u, but only in a simply-connected region, which is a region in which "*any closed curve can be continuously shrunk to a point without passing outside of the boundaries of the region*" [9]. For the proof of this I refer to [30] or [9, Chapter 10].

5.3 Derivation of Continuous Compatibility Conditions by a Method of Projections in Fourier Space

Because of the linearity, it is possible to use Fourier transform to define compatibility conditions in both 2-D and 3-D structures. In the space of Fourier images differentiation is replaced by multiplication by ω, and the problem is reduced to linear algebra, namely to finding an orthogonal projection. In fact, because there are three components and two arguments in 2-D, and six components and three arguments in 3-D, these extra components belong to some subspace. The orthogonal projection of this subspace should be a polynomial in ω. Once this polynomial is found, replacing each ω_i by $i\dfrac{\partial}{\partial x_i}$ would bring it back to the space of originals, and then it would be possible to check if this polynomial in Fourier space results in a continuum compatibility condition in the space of originals.

5.3.1 Two-Dimensional Case

The 2-D strain-displacement relations are given as

$$\varepsilon_{11} = u_{1,1}, \quad \varepsilon_{22} = u_{2,2}, \quad \varepsilon_{12} = \frac{1}{2}\left(u_{1,2} + u_{2,1}\right). \tag{5.10}$$

Then, using the Fourier transform, this differential relation between ε and u becomes

$$F(\varepsilon(u)) = \widehat{\varepsilon} = A(\omega) \cdot \widehat{u}, \tag{5.11}$$

where

$$A(\omega) = i \begin{pmatrix} \omega_1 & 0 \\ 0 & \omega_2 \\ \frac{\omega_2}{2} & \frac{\omega_1}{2} \end{pmatrix}.$$

Therefore, (5.11) becomes

$$\begin{pmatrix} \widehat{\varepsilon}_{11} \\ \widehat{\varepsilon}_{22} \\ \widehat{\varepsilon}_{12} \end{pmatrix} = i \begin{pmatrix} \omega_1 & 0 \\ 0 & \omega_2 \\ \frac{\omega_2}{2} & \frac{\omega_1}{2} \end{pmatrix} \cdot \begin{pmatrix} \widehat{u}_1 \\ \widehat{u}_2 \end{pmatrix}. \tag{5.12}$$

If there exists a matrix $P(\omega)$ such that

$$P(\omega) \cdot \widehat{\varepsilon} = P(\omega) \cdot A(\omega) \cdot \widehat{u} = 0, \tag{5.13}$$

then $P(\omega)$ represents the compatibility conditions in Fourier space. The following computation uncovers $P(\omega)$ as a projector:

$$\begin{aligned} \widehat{\varepsilon} &= A\widehat{u} \\ A^T\widehat{\varepsilon} &= A^T A\widehat{u} \\ \widehat{u} &= \left(A^T A\right)^{-1} A^T\widehat{\varepsilon} \\ A^{-1}\widehat{\varepsilon} &= \left(A^T A\right)^{-1} A^T\widehat{\varepsilon} \\ \widehat{\varepsilon} &= A\left(A^T A\right)^{-1} A^T\widehat{\varepsilon} \\ 0 &= \left(I - A\left(A^T A\right)^{-1} A^T\right)\widehat{\varepsilon} = \bar{P}\varepsilon, \end{aligned} \tag{5.14}$$

where $\bar{P}$ is the projector in question. Its entries are still not polynomials in ω, but multiplication of $\bar{P}$ by the determinant of $A^T A$ does produce a polynomial in ω:

$$P = \det\left(A^T A\right)\left(I - A\left(A^T A\right)^{-1} A^T\right). \tag{5.15}$$

$P\widehat{\varepsilon}$ represents compatibility conditions in Fourier space. Indeed, $P\widehat{\varepsilon}$ is a 3×1 matrix of rank 1, where each of the three equations is a multiple of the 2-D compatibility condition, once ω_i^j's are converted back to partial derivatives. In fact,

$$\omega_2^2\widehat{\varepsilon}_{11} + \omega_1^2\widehat{\varepsilon}_{22} - 2\omega_2\omega_1\widehat{\varepsilon}_{12} = 0. \tag{5.16}$$

Returning to the space of originals, (5.16) becomes

$$\frac{\partial^2}{\partial x_2^2}\varepsilon_{11} + \frac{\partial^2}{\partial x_1^2}\varepsilon_{22} - 2\frac{\partial^2}{\partial x_1 x_2}\varepsilon_{12} = \varepsilon_{11,22} + \varepsilon_{22,11} - 2\varepsilon_{12,12} = 0, \tag{5.17}$$

where $\omega_i^j \widehat{\varepsilon}_{kl} = \frac{\partial}{\partial x_i^j}\varepsilon_{kl}$, which is the 2-D continuum compatibility condition. The details of this computation is done on Maple, and can be seen in Appendix H on page 154.

5.3.2 Three-Dimensional Case

The approach in 3-D is quite similar, but now there are six, instead of three, strain-displacement relations:

$$\begin{aligned}
&\varepsilon_{11} = u_{1,1},\ \ \varepsilon_{22} = u_{2,2},\ \ \varepsilon_{33} = u_{3,3} \\
&\varepsilon_{12} = \frac{1}{2}\left(u_{1,2} + u_{2,1}\right),\ \ \varepsilon_{23} = \frac{1}{2}\left(u_{2,3} + u_{3,2}\right),\ \ \varepsilon_{31} = \frac{1}{2}\left(u_{3,1} + u_{1,3}\right).
\end{aligned} \tag{5.18}$$

Then this differentiation relation becomes

$$\begin{pmatrix} \widehat{\varepsilon}_{11} \\ \widehat{\varepsilon}_{22} \\ \widehat{\varepsilon}_{33} \\ \widehat{\varepsilon}_{12} \\ \widehat{\varepsilon}_{13} \\ \widehat{\varepsilon}_{23} \end{pmatrix} = i \begin{pmatrix} \omega_1 & 0 & 0 \\ 0 & \omega_2 & 0 \\ 0 & 0 & \omega_3 \\ \frac{\omega_2}{2} & \frac{\omega_1}{2} & 0 \\ \frac{\omega_3}{2} & 0 & \frac{\omega_1}{2} \\ 0 & \frac{\omega_3}{2} & \frac{\omega_2}{2} \end{pmatrix} \cdot \begin{pmatrix} \widehat{u}_1 \\ \widehat{u}_2 \\ \widehat{u}_3 \end{pmatrix}, \tag{5.19}$$

where

$$A(\omega) = i \begin{pmatrix} \omega_1 & 0 & 0 \\ 0 & \omega_2 & 0 \\ 0 & 0 & \omega_3 \\ \frac{\omega_2}{2} & \frac{\omega_1}{2} & 0 \\ \frac{\omega_3}{2} & 0 & \frac{\omega_1}{2} \\ 0 & \frac{\omega_3}{2} & \frac{\omega_2}{2} \end{pmatrix}.$$

$P\widehat{\varepsilon}$ gives us a 6×1 matrix, of rank 3, where the 6 equations are linear in $\widehat{\varepsilon}_{ij}$, just like in the 2-D case. The six equations obtained, after solving (5.14) for $\widehat{\varepsilon}_{ij}$, happen to be the six 3-D continuum compatibility conditions,

$$\varepsilon_{11,22} + \varepsilon_{22,11} - 2\varepsilon_{12,12} = 0 \tag{5.20}$$

$$\varepsilon_{22,33} + \varepsilon_{33,22} - 2\varepsilon_{23,23} = 0 \tag{5.21}$$

$$\varepsilon_{33,11} + \varepsilon_{11,33} - 2\varepsilon_{13,13} = 0 \tag{5.22}$$

$$\varepsilon_{11,23} + \varepsilon_{23,11} - \varepsilon_{13,12} - \varepsilon_{12,13} = 0 \tag{5.23}$$

$$\varepsilon_{22,13} + \varepsilon_{13,22} - \varepsilon_{12,23} - \varepsilon_{23,12} = 0 \tag{5.24}$$

$$\varepsilon_{33,12} + \varepsilon_{12,33} - \varepsilon_{23,13} - \varepsilon_{13,23} = 0, \tag{5.25}$$

when they are converted back from the Fourier space into the space of originals by changing ω_i's into $\frac{\partial}{\partial x_i}$'s. This computation was done on Maple, and can be seen in Appendix I on page 156.

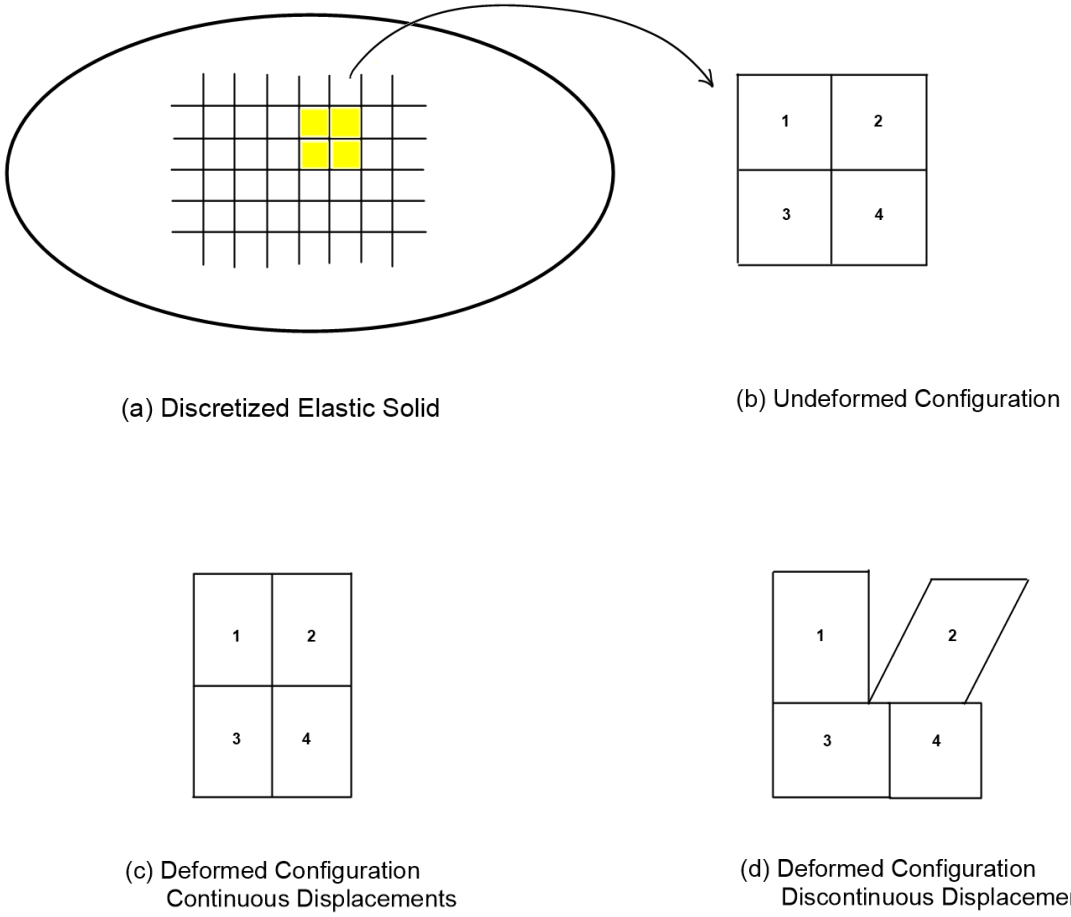

Figure 5.1: Here one can see the difference between a continuous deformation, that is the deformation which satisfies compatibility conditions, i.e., it stays in a compatible state after the deformation, versus the discontinuous deformation, the one that does not stay in a compatible state, and does not satisfy the compatibility conditions.

CHAPTER 6

DISCRETE COMPATIBILITY CONDITIONS

If deformation of the links is very small, then the displacements of the nodes compared to the lengths of the links are also small:

$$u_i[k] \ll L_{i,j}, \quad \text{for all } \{a_i, a_j\} \in E(m) \text{ and } k = 1, 2, 3 \tag{6.1}$$

where $u_i[k]$ is the displacement of the i-th node in the k-th direction. In this case, it is safe to linearize the strain components, as if a structure undergoes an infinitesimal deformation, which can be defined as instantaneous rate of change of the position of the nodes; or its velocity field at time $t = 0$.

In what follows, the deformation is assumed to be small enough to be safely linearized, and that is what is done in the first part of this chapter. It is then used to find the discrete compatibility conditions of various hexagons, including a general hexagon.

6.1 Linearization of Strain Components

Suppose $Q^d(S)$, with fixed parameters preventing the rigid-body motion, undergoes a flex. As was previously stated, the links between a_i and a_j are denoted by the vectors $l_{i,j} = a_j - a_i$, where $|l_{i,j}| = L_{i,j}$, for all connected i and j. Let $Q^d(S)$, $d = 2$ or 3, undergo a very small information, which can be safely linearized. The displacement of node i after some time t is denoted by $u_i(t)$, which represents the rate of change in the position, or the velocity, of the i-th node. In fact,

$$\left.\frac{d}{dt}\right|_{t=0} a_i(t) = u_i, \text{ for all } a_i \in S(n), \tag{6.2}$$

represents the instantaneous rate of change of the a_i's position, thus it is its instantaneous velocity. Note that $a_i(0) = a_i$ and $L_{i,j}(0) = L_{i,j}$ represent

the nondisplaced position of the i-th node, and the undeformed length of the $l_{i,j}$-th link, respectively.

The elongation of link $l_{i,j}$ is denoted by $\lambda_{i,j}$, and it is defined as the instantaneous rate of change of the length of $l_{i,j}$. That is,

$$\left.\frac{d}{dt}\right|_{t=0} |a_i(t) - a_j(t)| = \left.\frac{d}{dt}\right|_{t=0} |l_{i,j}(t)| = \left.\frac{d}{dt}\right|_{t=0} L_{i,j}(t) = \lambda_{i,j}. \tag{6.3}$$

Keeping in mind that a_i is $d \times 1$ matrix, for all $a_i \in S(n)$, the distance formula implies that

$$(a_i(t) - a_j(t))^T \cdot (a_i(t) - a_j(t)) = |l_{i,j}(t)|^2 = L_{i,j}^2(t), \tag{6.4}$$

for any $t \geq 0$. Because the deformation is small, it is linearized, or, more precisely, the LHS of (6.4) is differentiated with respect to t, as $t \to 0$, to obtain

$$\begin{aligned}\left.\frac{d}{dt}\right|_{t=0} (a_i - a_j)^T \cdot (a_i - a_j) &= (u_i - u_j)^T \cdot (a_i - a_j) + (u_i - u_j) \cdot (a_i - a_j)^T \\ &= 2\,(a_i - a_j) \cdot (u_i - u_j) \\ &= 2 l_{i,j} \cdot (u_i - u_j)\,. \end{aligned} \tag{6.5}$$

And if doing the same to the RHS of (6.4) gives

$$\left.\frac{d}{dt}\right|_{t=0} |l_{i,j}|^2 = 2\,|l_{i,j}| \cdot \lambda_{i,j} = 2 L_{i,j} \lambda_{i,j}. \tag{6.6}$$

Equating (6.5) with (6.6) gives

$$l_{i,j} \cdot (u_i - u_j) = L_{i,j} \lambda_{i,j}, \quad \text{for all linked } i, j, \tag{6.7}$$

which, in matrix form, is given in (3.1): $A \cdot U = \Lambda$. As mentioned in Chapter 3 on page 8, this is a system of M equations, where M depends on m and r, in $dn - r$ unknowns, d for each node displacement, u_i, of all n nodes, because each node has d degrees of freedom.

After the three parameters are fixed, in order to prevent the rigid-body motion, the system is left with $2n-3$ unknowns in m equations. The system is overdetermined if $m > 2n - 3$, and the number of compatibility equations is given by equation (3.2) for 2-D:

$$C_2 = m - 2n + 3.$$

6.2 Wagon-Wheel Condition

The goal is to obtain the compatibility condition of a general hexagon using linear algebra, as opposed to the technique used in Section 4.2 on page 38, that undergoes a small enough deformation that can be safely linearized, and use it to show that the discrete compatibility condition of a hexagonal lattice imposes the continuum compatibility condition, once the link lengths are taken to the continuum limit. This would show that it is possible to study the continuum case while studying the discrete hexagonal lattices.

6.2.1 Regular Hexagon

Consider a hexagonal structure given in Figure 6.1 on page 58.

In Section 4.2 on page 38, it is computed that the one compatibility condition of any regular hexagon is given as

$$\sum_{j=1}^{6} (\lambda_j - \mu_j) = 0, \tag{6.8}$$

where the λ_j represent the elongations of the radial links, while the μ_j represent the elongations of the circumferential links. The same results should be obtained using linear algebra in this chapter.

The coordinates of the 7 nodes of this regular hexagon are:

$$a_0 = (0,0), a_1 = (1,0), a_2 = \left(\frac{1}{2}, \frac{\sqrt{3}}{2}\right), a_3 = \left(-\frac{1}{2}, \frac{\sqrt{3}}{2}\right),$$

$$a_4 = (-1,0), a_5 = \left(-\frac{1}{2}, -\frac{\sqrt{3}}{2}\right), a_6 = \left(\frac{1}{2}, \frac{\sqrt{3}}{2}\right).$$

As usual, the links, and there are 12 of them, between a_i and a_j are denoted by the vectors $l_{i,j} = a_j - a_i$, where $|l_{i,j}| = L_{i,j} = 1$, for all connected i and j.

Let this hexagon undergo a small deformation. Node displacements are denoted by u_i, while the link elongations are $\lambda_{i,j}$. The rigid body motion is prevented by fixing the middle node,

$$u_0 = (0,0) = 0, \tag{6.9}$$

and by setting the total angular momentum about u_0 equal to zero,

$$\sum_{i=1}^{n-1} a_i \times u_i = \sum_{i=1}^{n-1} (a_i[1] \cdot u_i[2] - a_i[2] \cdot u_i[1]) = 0, \tag{6.10}$$

where $a_i[j]$ and $u_i[j]$, $j = 1, 2$, represent the j-th coordinate of a_i and u_i, respectively. The equation (6.9) solves for two unknowns, namely $u_0[1] = 0$ and $u_0[2] = 0$, while equation (6.10) adds to the 12 equations for the 12 links, (6.7). The linear system now has 13 equations,

$$\begin{aligned} \sum_{i=1}^{6} (a_i[1] \cdot u_i[2] - a_i[2] \cdot u_i[1]) = 0, \\ (u_i - u_j) \cdot l_{i,j} = \lambda_{i,j}, \quad \text{for all linked } i, j \end{aligned} \tag{6.11}$$

in 12 unknowns, that are the displacements of the remaining 6 nodes. This overdetermined system with linearly independent equations gives rise to the following solution

$$\sum_{i=1}^{6} \lambda_{i,0} = \sum_{i=1}^{6} \lambda_{i+1,i}, \tag{6.12}$$

where i is modulo six. This is equivalent to (6.8): the LHS of (6.12) is the sum of all radial (inner) links elongations, and the RHS of (6.12) is the sum of all circumferential (boundary or outer) link elongations.

6.2.2 Affine-Regular Hexagon

A hexagon obtained by an affine transformation of a regular hexagon is called an affine-regular hexagon, and the one given in Figure 6.2 on page 58, that has following coordinates,

$$a_0 = (0,0), a_1 = (1,0), a_2 = (1,1), a_3 = (0,1),$$

$$a_4 = (-1,0), a_5 = (-1,-1), a_6 = (0,-1).$$

is just one specific example of it. The links of this hexagon are of lengths 1 and $\sqrt{2}$. The same linear algebra procedure produces the following compatibility condition

$$\sum_{j=1}^{6} L_{j,0} \cdot \lambda_{j,0} = \sum_{j=1}^{6} L_{j+1,j} \cdot \lambda_{i+1,i}, \tag{6.13}$$

where $j = 1$ when $j = 6$. This is *not* the same equation that was obtained in Section 4.2 on page 38 for general polygonal triangulations; rather it is, in some way, the opposite of that condition in that the link elongations are multiplied with their own undeformed lengths, whereas for regular polygons they are multiplied by the undeformed lengths of the other group of links (if it is a circumferential link its elongation is multiplied by the corresponding undeformed radial link, and vice versa).

A generic affine-regular hexagon,[1] which is *any* generic hexagon that is related to the regular hexagon by an affine transformation, has the following general coordinates of the nodes:

$$a_0 = (0,0), a_1 = (1,0), a_2 = (v,w), a_3 = (v-1,w),$$

$$a_4 = (-1,0), a_5 = (-v,-w), a_6 = (1-v,-w),$$

where $v, w \in \mathbb{R} - \{0\}$.

The same analysis yields the same compatibility condition as with the special-case affine-regular hexagon, namely (6.13), but the lengths are 1, $\sqrt{(1-v)^2 + w}$, and $\sqrt{v^2 + w^2}$, which reduces to 1 and $\sqrt{2}$ if $v = w = 1$.

6.2.3 The General Hexagon

The coordinates of the nodes of a general hexagon are

$$a_0 = (0,0), a_1 = (1,0), a_2 = (p,q), a_3 = (r,s),$$

[1] Generic in a sense that strict triangle inequalities are required; so those hexagons that have their links aligned are not in the set of generic hexagons. Because the set of nongeneric hexagons is of measure zero, if one were to randomly pick a hexagon, chances of it being generic are 100%.

$$a_4 = (t, u), a_5 = (v, w), a_6 = (x, y),$$

where $p, q, r, s, t, u, v, w, x, y \in \mathbb{R} - \{0\}$ (Figure 6.4 on page 59). However, in order for a structure with these coordinates for its nodes to fit a definition of a generic hexagon, stricter restrictions on $p, q, \ldots, y$ are needed. However, for what follows no stricter restrictions are needed, as all the calculations and results hold even if the structure does not look anything like a hexagon.

The same linear algebra procedure gives the following result for the compatibility condition of a general hexagon:

$$\begin{aligned}
&(-wx + yv)(-uv + wt)(-st + ur)(-y + yp + q - qx)(sp - rq)\lambda_{1,0} \\
&-\sqrt{p^2 + q^2}(-wx + yv)y(-uv + wt)(-st + ur)(-rq + q - s + sp)\lambda_{2,0} \\
&-\sqrt{(r^2 + s^2)}qy(-wx + yv)(-uv + wt)(-rq + tq - st - up + sp + ur)\lambda_{3,0} \\
&-\sqrt{t^2 + u^2}q(-wx + yv)y(vs - uv - st - rw + wt + ur)(sp - rq)\lambda_{4,0} \\
&-\sqrt{v^2 + w^2}q(-ty - wx + yv + wt + ux - uv)y(-st + ur)(sp - rq)\lambda_{5,0} \\
&+\sqrt{x^2 + y^2}q(w - y + yv - wx)(-uv + wt)(-st + ur)(sp - rq)\lambda_{6,0} \\
&+\sqrt{p^2 - 2p + 1 + q^2}(-wx + yv)y(-uv + wt)(-st + ur)(sp - rq)\lambda_{2,1} \\
&+\sqrt{r^2 - 2rp + p^2 + s^2 - 2sq + q^2}q(-wx + yv)y(-uv + wt)(-st + ur)\lambda_{3,2} \\
&+\sqrt{t^2 - 2tr + r^2 + u^2 - 2us + s^2}q(-wx + yv)y(-uv + wt)(sp - rq)\lambda_{4,3} \\
&+\sqrt{v^2 - 2vt + t^2 + w^2 - 2wu + u^2}q(-wx + yv)y(-st + ur)(sp - rq)\lambda_{5,4} \\
&+\sqrt{x^2 - 2xv + v^2 + y^2 - 2yw + w^2}q(-uv + wt)y(-st + ur)(sp - rq)\lambda_{6,5} \\
&-\sqrt{x^2 - 2x + 1 + y^2}q(-wx + yv)(-uv + wt)(-st + ur)(sp - rq)\lambda_{6,1} \\
&= 0. \quad (6.14)
\end{aligned}$$

It is still a weighted sum of the elongations, but the weights are quite different now. In an affine hexagon, all of the coefficients of λ's were equal to $\pm\lambda$ times their respective lengths, where the sign indicated whether the link was radial or circumferential. Now they are much more complicated. In fact, it has been observed that they are all areas of parallelograms and triangles.

The situation is much simpler with circumferential links than with radial links. The first circumferential link's elongation is $\lambda_{2,1}$, and its coefficient is

$$\sqrt{p^2-2p+1+q^2}(-wx+yv)y(-uv+wt)(-st+ur)(sp-rq).$$

The first term, $\sqrt{p^2-2p+1+q^2}$, is the length of an undeformed link connecting $(1,0)$ and (p,q), that is $|l_{2,1}| = L_{1,2} = \sqrt{p^2-2p+1+q^2}$. The other five terms are the areas of the five parallelograms that *do not* enclose $l_{2,1}$. Indeed, they are the cross products of the neighboring radial links, with the only cross product missing being $l_{2,0} \times l_{1,0}$.

The same is true for all the other circumferential links: for example, every term of any $\lambda_{i+1,i}$, where $i = 1, \ldots, 6$ and $i = 1$ when $= 6$ (except for the first one, which is the length of $l_{i,0}$) is given by $l_{i+2,0} \times l_{i+1,0}$, where $i = 1, \ldots, 5$ and $i = 1$ when $i = 6$, so the only cross product missing is $l_{i+1} \times l_i$.

The radial links are somewhat different, which can be seen by the *longer* terms. For the coefficient of $\lambda_{4,0}$, say, the four *shorter* terms are the areas of the four parallelograms that do not have $l_{4,0}$ as one of their sides, while the *longer* term is the half-area of the triangle with the nodes a_3, a_4, a_5, such that one of the triangle links crosses the link $l_{5,0}$. In fact,

$$\frac{1}{2}A\left(\Delta a_3 a_4 a_5\right) = \begin{vmatrix} r & s & 1 \\ t & u & 1 \\ v & w & 1 \end{vmatrix} = (vs - uv - st - rw + wt + ur).$$

The pattern is exactly the same for all the other radial links. These parallelograms can be seen in Figure 6.5 on page 60.

The discrete compatibility conditions found in this chapter for different types of hexagons are called the wagon-wheel conditions, and it is shown in the next chapter that, when they are taken to the continuum limit, they impose the continuum compatibility condition, if some regularities on the strains of a continuum are assumed.

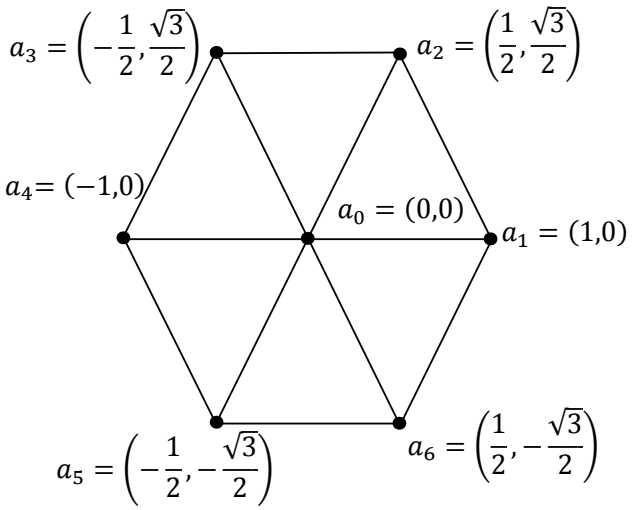

Figure 6.1: Regular hexagonal structure with $n = 7$ and $m = 12$.

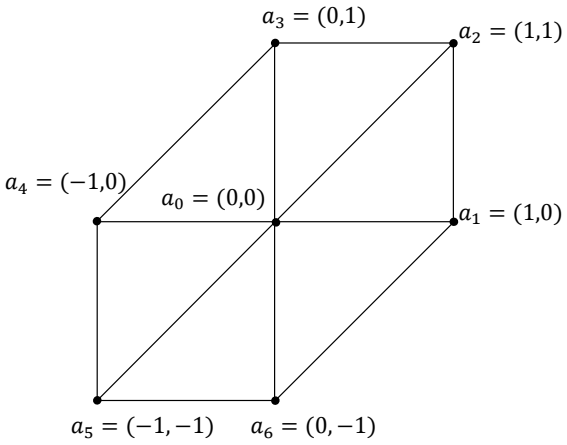

Figure 6.2: Some links are of length 1, some of length $\sqrt{2}$.

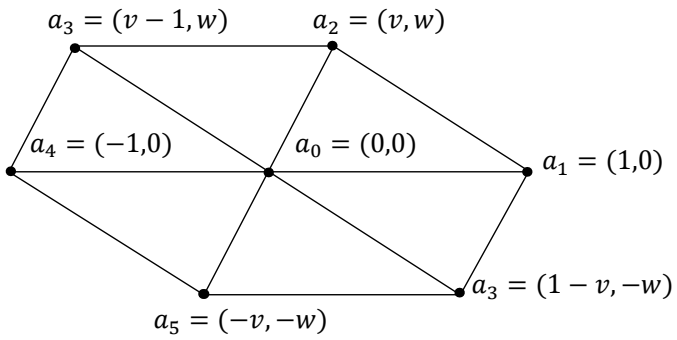

Figure 6.3: This is a typical affine-regular hexagon.

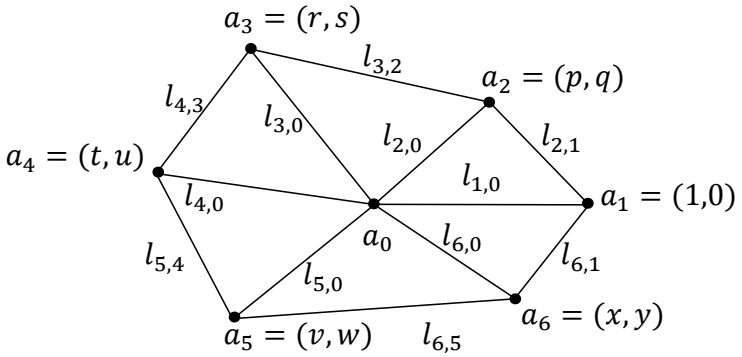

Figure 6.4: This is a typical general hexagon.

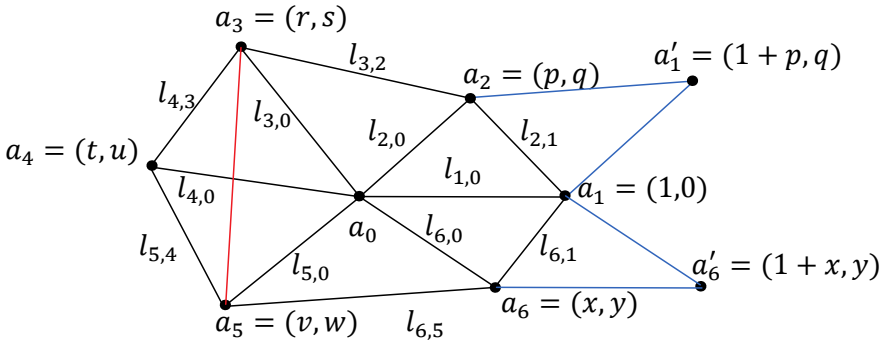

Figure 6.5: The parallelograms $a_0a_1a_1'a_2$ and $a_0a_1a_6'a_6$ are just two of six whose areas are coefficients of the obtained compatibility condition for the discrete regular hexagon, given in (6.14).

CHAPTER 7

DISCRETE TO CONTINUUM

Let Q^R be any regular hexagon, and let it undergo a deformation. In the preceding section it was shown that such a hexagon, for deformations sufficiently small, satisfies the wagon-wheel condition given in (6.13). Let Q^R_δ, a regular δ-hexagon, where δ is an arbitrarily small positive number, be Q^R scaled down by δ. Because of the linearity of the wagon-wheel condition the scaling can be factored out, and, therefore, the wagon-wheel condition holds for any sized hexagon; in particular, it holds for Q^R_δ.

Now imagine a continuum approximated by Q^R_δ, or by a regular-hexagonal triangulation, where each regular hexagon has links of size $\delta > 0$. The question is, if $\delta \to 0$, can, then, the 2-D continuum compatibility condition,

$$\frac{\partial^2}{\partial x_2^2}\varepsilon_{11} + \frac{\partial^2}{\partial x_1^2}\varepsilon_{22} - 2\frac{\partial^2}{\partial x_1 x_2}\varepsilon_{12} = \varepsilon_{11,22} + \varepsilon_{22,11} - 2\varepsilon_{12,12} = 0, \tag{7.1}$$

where ε_{ij} represent the strains of some continuum, be deduced from the wagon-wheel condition of Q^R_δ? The answer is yes. As a first step towards showing this fact, a lemma that expresses the change in lengths of the sides of an arbitrarily small hexagon, Q_δ, in terms of the strain matrix of a deformation, is developed next.

7.1 The Lemma

We are interested in what the restrictions on the lengths of hexagon links — hexagonal compatibility condition, which we named the wagon-wheel condition — can tell us about the strains of a continuum. In order to study this, first the link elongations need to be related to strains. This is done in the following lemma, link by link.

Lemma 7.1. *Let $a_i, a_j \in \mathbb{R}^2$, and let $\varphi : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}$ be a deformation such that $\varphi(x, 0) = x$. Then*

$$\lambda_{j,i} = \frac{d}{dt}\bigg|_{t=0} d\left(\varphi(a_j, t), \varphi(a_i, t)\right) = \frac{1}{|a_j - a_i|} \int_0^1 (a_j - a_i)^T \varepsilon(\gamma(s))(a_j - a_i)\, ds, \quad (7.2)$$

where ε is the strain matrix, and $\gamma(s) = a_i - s(a_i - a_j)$ is a linear parameterization of the link between a_i and a_j.

Proof. Because φ is a flow, its derivative is velocity; and, in particular,

$$u(x) = \frac{d}{dt}\varphi(x, t) = \text{velocity field at } t = 0. \quad (7.3)$$

Now, let u denote the instantaneous (at $t = 0$) rate of change of the distance of endpoints under flow. Then,

$$\begin{aligned}
\lambda_{j,i} &= \frac{d}{dt}\bigg|_{t=0} d\left(\varphi(a_j, t), \varphi(a_i, t)\right) \\
&= \frac{d}{dt}\bigg|_{t=0} \sqrt{[\varphi(a_j, t) - \varphi(a_i, t)]^T \cdot [\varphi(a_j, t) - \varphi(a_i, t)]} \\
&= \frac{(\varphi(a_j, t) - \varphi(a_i, t))^T \cdot (u(a_j) - u(a_i))}{\sqrt{[\varphi(a_j, t) - \varphi(a_i, t)]^T \cdot [\varphi(a_j, t) - \varphi(a_i, t)]}}\bigg|_{t=0} \\
&= \frac{(a_j - a_i)^T \cdot (u(a_j) - u(a_i))}{|a_j - a_i|}.
\end{aligned}$$

Let $\gamma(s) = a_i - s(a_i - a_j)$, where $s \in [0, 1]$. Then,

$$\begin{aligned}
u(a_j) - u(a_i) &= \int_0^1 \frac{d}{ds}\, u(\gamma(s))\, ds \\
&= \int_0^1 \nabla u(\gamma(s)) \cdot \dot{\gamma}(s)\, ds
\end{aligned}$$

But $\dot{\gamma}(s) = a_j - a_i$, so

$$\lambda_{j,i} = \frac{1}{|a_j - a_i|} \int_0^1 (a_j - a_i)^T \nabla u(\gamma(s))(a_j - a_i)\, ds$$

$$= \frac{1}{|a_j - a_i|} \int_0^1 (a_j - a_i)^T \varepsilon(\gamma(s))(a_j - a_i)\, ds,$$

because $\varepsilon = \frac{1}{2}\left(\nabla u + (\nabla u)^T\right)$, and the integrand is a quadratic. In fact,

$$\begin{aligned}(a_j - a_i)^T \varepsilon (a_j - a_i) &= \frac{1}{2}(a_j - a_i)^T \left(\nabla u + (\nabla u)^T\right)(a_j - a_i) \\ &= \frac{1}{2}\left((a_j - a_i)^T \nabla u (a_j - a_i) + (a_j - a_i)^T (\nabla u)^T (a_j - a_i)\right) \\ &= (a_j - a_i)^T \nabla u (a_j - a_i)\end{aligned}$$

because $A = (a_j - a_i)^T \nabla u (a_j - a_i)$ is a scalar, which implies that $A = A^T$. □

Now, if the velocity field is given by $u(x) = \frac{d}{dt}\big|_{t=0} \varphi(x,t)$, where $\varphi(x,t)$ is a local deformation of the plane, then the usual 2-D continuum compatibility condition proof applies (see Section 5.2 on page 44).

However, if the only assumption is that the vector field u satisfies the linear strain equation

$$\varepsilon_{i,j} = \frac{1}{2}\left(\nabla u + (\nabla u)^T\right), \tag{7.4}$$

i.e., it is a C^3 solution to (7.4), and that it behaves well with respect to the approximation — if the nodes of hexagons moved by u satisfy the wagon-wheel condition — then the 2-D continuum compatibility condition can still be deduced. In other words, the approximate discrete compatibility condition implies the 2-D continuum compatibility condition, and this is formally stated and proved in the next section.

7.2 The Main Theorem

The compatibility equations of the linearized discrete problem on a hexagonal grid give information on the underlying planar deformation. What follows is a theorem that shows that a Taylor's expansion of the wagon wheel condition in terms of the strains gives the continuum compatibility conditions as the leading term.

Theorem 7.2 (Discrete to Continuum)**.** *Let Ω be an open convex set, and let δ be small enough so that a δ-hexagon that satisfies the wagon-wheel condition can fit entirely inside the compact subset of Ω. Assume there is background planar deformation that moves the nodes of the hexagon, and let the strains be $C^r(\Omega)$ functions, $r \geq 3$, satisfying the following strain-displacement relation*

$$\varepsilon_{i,j} = \frac{1}{2}\left(\nabla u + (\nabla u)^T\right), \tag{7.5}$$

by which the strains of the plane and the displacements of the nodes are related. Then, as δ goes to zero, the wagon-wheel condition of the hexagon gives rise to the following Taylor's expression about the center of the hexagon in terms of the strains

$$0 = \frac{3}{4}\left(\varepsilon_{11,22} + \varepsilon_{22,11} - 2\varepsilon_{12,12}\right)\delta^2 + 0 \cdot \delta^3 + o(\delta^3). \tag{7.6}$$

Proof. Because the δ-hexagon is small enough to fit entirely inside the compact subset of Ω, Lemma 7.1 on page 62 can be used to express the elongation of each link of the hexagon in terms of the strains. If the strains are C^r, $r \geq 3$, they can be approximated to within small error by Taylor polynomials of degree $r \geq 3$ around the center of this δ-hexagon. Indeed,

$$\varepsilon = \begin{pmatrix} \varepsilon_{11} & \varepsilon_{12} \\ \varepsilon_{12} & \varepsilon_{22} \end{pmatrix} = \begin{pmatrix} M & N \\ N & P \end{pmatrix} \tag{7.7}$$

where

$$M = \sum_{i=0}^{r}\sum_{j=0}^{i}\binom{i}{j}\frac{m_{i,j}x^j y^{i-j}}{i!} + o\left(|(x,y)|^r\right) \tag{7.8}$$

$$N = \sum_{i=0}^{r}\sum_{j=0}^{i}\binom{i}{j}\frac{n_{i,j}x^j y^{i-j}}{i!} + o\left(|(x,y)|^r\right) \tag{7.9}$$

$$P = \sum_{i=0}^{r}\sum_{j=0}^{i}\binom{i}{j}\frac{p_{i,j}x^j y^{i-j}}{i!} + o\left(|(x,y)|^r\right). \tag{7.10}$$

In $m_{i,j}$, i represents the differentiation degree, while j says how many times it is taken with respect to the first variable. Therefore,

$$m_{2,0} = \varepsilon_{11,22}, \qquad m_{2,1} = \varepsilon_{11,12}, \qquad m_{2,2} = \varepsilon_{11,11},$$

$$n_{2,0} = \varepsilon_{12,22}, \qquad n_{2,1} = \varepsilon_{12,12}, \qquad n_{2,2} = \varepsilon_{12,11},$$

$$p_{2,0} = \varepsilon_{22,22}, \qquad p_{2,1} = \varepsilon_{22,12}, \qquad p_{2,2} = \varepsilon_{22,11}.$$

Lemma 7.1 on page 62 now implies that, using this close approximation of a strain at any point in a compact subset of Ω, it is possible to compute the elongation, $\lambda_{i,j}$, of a link $l_{i,j}$ with endpoints a_i and a_j.

Because Lemma 7.1 on page 62 holds for a pair of points that are of any distance apart, it, in particular, holds for all connected nodes of the δ-hexagon. Therefore, it can be used to represent the elongation of each link of the δ-hexagon in terms of a strain matrix, which is represented by the integral in Lemma 7.1 on page 62. Then, the question is what restrictions on the strains are imposed by the wagon-wheel condition of the hexagon with its link elongations expressed in terms of the strains.

Each radial link of Q_δ^R has the following parameterization:

$$\gamma(s) = \delta \cdot a_0 + s \cdot (a_k - a_0)\,, \text{ for all } k = 1, \ldots, 6 \text{ and } s \in [0, \delta],$$

while each circumferential link is parameterized as follows:

$$\gamma(s) = \delta \cdot a_k + s \cdot (a_{k+1} - a_k)\,, \text{ for all } k = 1, \ldots, 6 \text{ and } s \in [0, \delta],$$

where $a_7 = a_1$. The parametrization is embedded in (7.7), and the strain matrix becomes:

$$\varepsilon(\gamma(s)) = \begin{bmatrix} M(\gamma(s)) & N(\gamma(s)) \\ N(\gamma(s)) & P(\gamma(s)) \end{bmatrix}, \tag{7.11}$$

while the lemma's equation is

$$\lambda_{i,j} = \frac{1}{\delta\,|a_j - a_i|} \int_0^\delta (a_j - a_i)^T \begin{bmatrix} M(\gamma(s)) & N(\gamma(s)) \\ N(\gamma(s)) & P(\gamma(s)) \end{bmatrix} (a_j - a_i)\, ds. \tag{7.12}$$

After applying (7.12) to each link of Q_δ^R, using the Maple symbolic-algebra system (the code is given in Appendix C on page 117) each $\lambda_{i,j}$ for all linked i, j, becomes expressed in terms of the strains. If these strain-expressed $\lambda_{i,j}$ are plugged into equation (6.13) (the wagon-wheel condition), the following condition on the strains is obtained

$$0 = \frac{3}{4}\left(\varepsilon_{11,22} + \varepsilon_{22,11} - 2\varepsilon_{12,12}\right)\delta^2 + 0 \cdot \delta^3 + o(\delta^3), \tag{7.13}$$

as δ goes to zero. □

Notice that the assumptions of Theorem 7.2 on page 64 are quite reasonable. For example, the assumption that a hexagon is small enough to fit inside the open convex set Ω, whatever its size, is reasonable because the wagon-wheel condition holds for any sized hexagon.

The assumption that the strains are C^3 is necessary in order to get enough terms from the Taylor polynomial, because the continuum compatibility condition contains second-order differentiation on strains, and in order to show that the coefficient of δ^3 vanishes.

Furthermore, the error terms do not contribute to the lowest-degree term of the polynomial on the right-hand side of (7.13). In fact, no matter what degree of a Taylor polynomial is chosen for the strains, the leading coefficient of that polynomial is always the continuum compatibility condition. In addition, these error terms are uniformly bounded because the δ-hexagon sits inside the compact subset of the open set Ω, so it can safely zoom-in its center.

The immediate consequence of the result obtained from Theorem 7.2 on page 64 is that the continuum compatibility condition holds for the point at the center of the δ-hexagon. In fact, as δ goes to zero, the error term is controlled and vanishes faster than the leading coefficient of (7.6); so, after dividing (7.6) by δ^2, and then letting $\delta \to 0$, the following holds

$$\varepsilon_{11,22} + \varepsilon_{22,11} - 2\varepsilon_{12,12} = 0, \tag{7.14}$$

which is the continuum compatibility condition.

Moreover, it was found that the continuum compatibility condition is deduced from the wagon-wheel conditions of a *general hexagon*, as well, as long as it is centered at the point in question. The derivation of those conditions can be found in Section 6.2.3 on page 55. This was verified by numerical experiments shown in Appendix D on page 124.

In general, the continuum compatibility condition holds at *any* point in a continuum if a countable sequence of regular hexagonal grids that get finer and finer (δ gets smaller and smaller), where each regular hexagon satisfies

the wagon-wheel condition, is used. Namely, because the continuum compatibility condition is the leading coefficient in the Taylor's expansion about *any* point *inside* the regular hexagon, not just the center (the code that shows this is in Appendix C on page 117), then the continuum compatibility condition holds at a point in question because it must lie inside one such hexagon that belongs to one of the grids from the sequence.

CHAPTER 8

THREE-DIMENSIONAL STRUCTURES: CUBOCTAHEDRON

The natural move from hexagonal lattices and 2-D compatibility conditions was the study of 3-D objects and their compatibility conditions. In close-packed spheres, its closest packing is a cuboctahedron. Namely, *"the closest packing of equal spheres around a nucleus of equal size gives the dymaxion or cuboctahedron. The nuclear sphere is surrounded by twelve spheres, each touching four neighbors in addition to the nucleus."* [25]. This is very well illustrated in Figure 8.1 on page 72, courtesy of [25].

If an infinite space is filled with equal-sized balls, one can imagine overlapping cuboctahedrons as the only units filling out the space. The center of each cuboctahedron is connected with twelve links. Because each of these links is connected to another center of a different cuboctahedron, these are being counted twice, so there are six links for each node, and each node has three degrees of freedom. Therefore, a regular cuboctahedron should have three compatibility conditions, according to equation (3.2).

The nodes of a regular cuboctahedron with links of length 1 have the following coordinates:

$$a_0 = (0,0,0), a_1 = (1,0,0), a_2 = \left(\frac{1}{2}, \frac{\sqrt{3}}{2}, 0\right), a_3 = \left(-\frac{1}{2}, \frac{\sqrt{3}}{2}, 0\right),$$

$$a_4 = (-1,0,0), a_5 = \left(-\frac{1}{2}, -\frac{\sqrt{3}}{2}, 0\right), a_6 = \left(\frac{1}{2}, -\frac{\sqrt{3}}{2}, 0\right),$$

$$a_7 = \left(\frac{1}{2}, \frac{\sqrt{3}}{2}, \frac{\sqrt{6}}{3}\right), a_8 = \left(-\frac{1}{2}, \frac{\sqrt{3}}{2}, \frac{\sqrt{6}}{3}\right), a_9 = \left(0, -\frac{\sqrt{3}}{2}, \frac{\sqrt{6}}{3}\right),$$

$$a_{10} = \left(\frac{1}{2}, -\frac{\sqrt{3}}{2}, -\frac{\sqrt{6}}{3}\right), a_{11} = \left(0, -\frac{\sqrt{3}}{2}, -\frac{\sqrt{6}}{3}\right), a_{12} = \left(-\frac{1}{2}, -\frac{\sqrt{3}}{2}, -\frac{\sqrt{6}}{3}\right).$$

In order to study the global rigidity and compatibility conditions of a cuboctahedron, the minimum number of parameters has to be fixed in order to prevent its rigid-body motion. The same parameters that were fixed when starting a hexagon are fixed for a cuboctahedron. Namely, middle node is fixed,

$$u_0 = (0, 0, 0) = 0, \tag{8.1}$$

and the total angular momentum about u_0,

$$\sum_{i=1}^{12} a_i \times u_i = 0. \tag{8.2}$$

Or in coordinate form,

$$\begin{aligned} f_1 &= \sum_{i=1}^{12} (a_i[1] \cdot u_i[2] - a_i[2] \cdot u_i[1]) = 0, \\ f_2 &= \sum_{i=1}^{12} (a_i[2] \cdot u_i[3] - a_i[3] \cdot u_i[2]) = 0, \\ f_3 &= \sum_{i=1}^{12} (a_i[3] \cdot u_i[1] - a_i[1] \cdot u_i[3]) = 0. \end{aligned} \tag{8.3}$$

The following equations represent the relation between elongations and deformations:

$$(u_i - u_j) \cdot l_{i,j} = \lambda_{i,j} \text{ for all linked } i, j. \tag{8.4}$$

As expected, there are thirty-six equations, which is the number of links: twelve radial links, and twenty-four surface links. This can be seen by the following: each of the twelve loose nodes is connected to the fixed, middle node, $(0, 0, 0)$, with one link; these are the twelve radial links. Next, there are six links in the center plane, three in each upper and lower planes (the triangles), and six each for connecting upper and lower planes with the center: $6 + 2 \cdot 3 + 2 \cdot 6 = 24$.

Because there are thirteen nodes, and each of these nodes has three degrees of freedom, a cuboctahedron has thirty-nine degrees of freedom in total. However, one node is completely fixed, taking three degrees of

freedom away, and the angular momentum is fixed as well, taking another three degrees of freedom away. In all, six degrees of freedom are occupied by the fixed parameters, resulting in thirty-three degrees of freedom for a fixed cuboctahedron. Therefore, $m = 36$, $d = 3$, $r = 6$, and $n = 13$ implies that

$$C_3 = 36 + 6 - 3 \cdot 13 = 3,$$

according to (3.2). This means that a regular cuboctahedron has three compatibility conditions, as was geometrically established earlier using an infinite space filled with equal-sized balls.

The thirty-six equations from (8.4), plus the three from (8.2), make up thirty-nine equations in thirty-six unknowns — an overdetermined system, that can be given in the matrix form (equation (3.1)):

$$A \cdot U = \Lambda,$$

whose solution should give us three compatibility conditions.

However, the result is four *independent* equations, which implies extra dependency among the thirty-nine equations. In other words, the kernel of A is not trivial, which implies infinitesimal flexibility. In fact, the symmetry of a cuboctahedron gives rise to an additional compatibility condition. But, if the symmetry is broken by perturbing the cuboctahedron's nodes by any nonzero amount, the system returns only three compatibility conditions, as is demonstrated by a numerical experiment done in Appendix J on page 159.

8.1 Geometrical Explanation

The most interesting finding is that the four compatibility conditions of a cuboctahedron represent the four intersecting hexagons every cuboctahedron has, that can be seen in Figure 8.3 on page 73.

The equations of these four wagon-wheel conditions are the following:

$$\begin{aligned}
\lambda_{0,1}+\lambda_{0,7}+\lambda_{0,8}+\lambda_{0,4}+\lambda_{0,12}+\lambda_{0,10} &= \lambda_{1,10}+\lambda_{1,7}+\lambda_{7,8}+\lambda_{4,8}+\lambda_{4,12}+\lambda_{10,12}\\
\lambda_{0,1}+\lambda_{0,2}+\lambda_{0,3}+\lambda_{0,4}+\lambda_{0,5}+\lambda_{0,6} &= \lambda_{1,2}+\lambda_{2,3}+\lambda_{3,4}+\lambda_{4,5}+\lambda_{5,6}+\lambda_{1,6}\\
\lambda_{0,3}+\lambda_{0,8}+\lambda_{0,9}+\lambda_{0,6}+\lambda_{0,10}+\lambda_{0,11} &= \lambda_{3,8}+\lambda_{8,9}+\lambda_{6,9}+\lambda_{6,10}+\lambda_{9,11}+\lambda_{3,11}\\
\lambda_{0,2}+\lambda_{0,7}+\lambda_{0,9}+\lambda_{0,5}+\lambda_{0,12}+\lambda_{0,11} &= \lambda_{2,7}+\lambda_{7,9}+\lambda_{5,9}+\lambda_{5,12}+\lambda_{5,11}+\lambda_{2,11}. \qquad (8.5)
\end{aligned}$$

After the symmetry is broken, the four planar compatibility conditions are replaced by the three spatial ones, and infinitesimal rigidity is achieved. These three compatibility conditions do not seem to represent anything significant geometrically, and they vary depending on the perturbed amount.

The fact is that a regular cuboctahedron is not infinitesimally rigid — a structure is said to be infinitesimally rigid if the kernel of the homogeneous, linearized, prescribed-length problem consists only of rigid motions. If a structure is fixed, like in our case, in order to disable the rigid-body motion, an infinitesimally rigid structure has a trivial kernel. A regular cuboctahedron in a linearized setting does not have a trivial kernel, implying the infinitesimal flexibility. Namely, setting each elongation to zero and solving those 39 equations in 36 unknowns again does not result into each displacement being equal to zero. This suggests nontrivial velocity, hence the infinitesimal flexibility.

Geometrically, this can be visualized by fixing each node around the middle node, and observing its motion. In a linearized setting a vertical motion of the middle node is possible, and this is precisely that additional degree of freedom that gives rise to four compatibility conditions.

However, interestingly enough, if the nodes are perturbed by any nonzero amount, the symmetry is broken, resulting in the three nonplanar compatibility conditions, as expected.

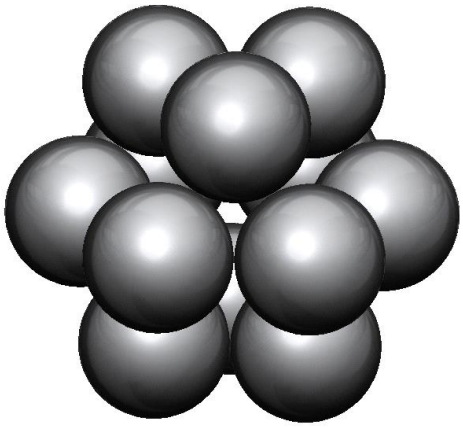

Figure 8.1: The closest packing of spheres resembles a cuboctahedron.

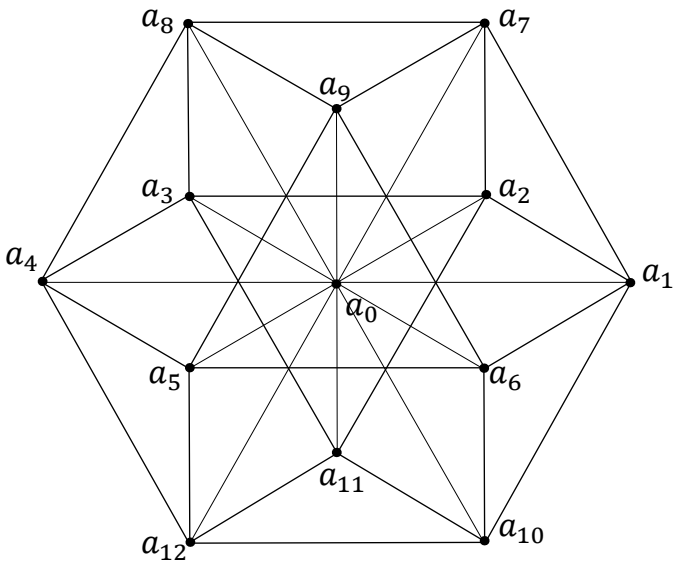

Figure 8.2: This is a regular cuboctahedron.

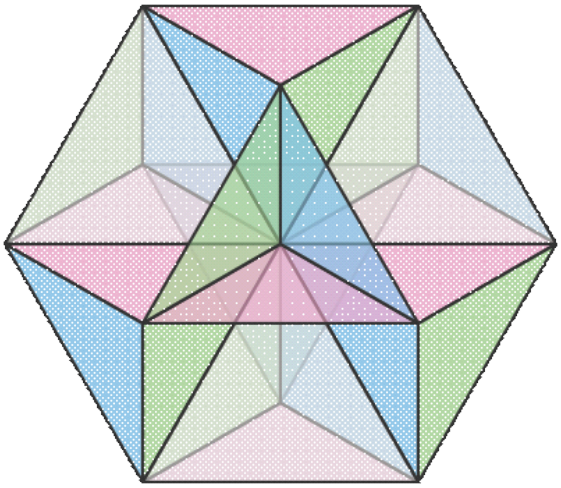

Figure 8.3: The four hexagons are clearly visible here.

CHAPTER 9

MECHANICS AND APPLICATION TO DAMAGE

This chapter deals with analysis of different Q_τ^2's, under various loadings and initial conditions, using computer programming. Deformation is assumed to be small enough to be linearized. The vertical displacements are ignored for horizontal links when solving the stiffness matrix of a structure. There are plenty of materials that undergo small deformation, such as wood, concrete, or steel, making these linear approximations applicable.

The literature that aided in understanding the concepts of damaged lattices and their behavior under the applied loading include the works of A. Cherkaev and L. Zhornitskaya, [6] and [8], as well as works by A. Cherkaev, V. Vinogradov, and S. Leelavanichkul, [5], and also [4] and [3] by A. Cherkaev and S. Leelavanichkul. The class notes by David J. Raymond, [26], proved useful in understanding the notion of stress and kinematics in continuum mechanics, as did the classic work by James Clerk Maxwell [23].

9.1 Notation and Formulation of the Problem

Gilbert Strang's book [31], §2.4 (Structures in Equilibrium) in particular, proved useful in building the Maple code for the experiments in this chapter. The same steps in building the 2-D equilibrium equations and the stiffness matrix that are used in §2.4 of [31] are used in this document. The "structural equilibrium" equations, with small notational changes from those used in §2.4 of [31], is the following:

U : Displacements at the free nodes ($2n - r$ degrees of freedom).

$AU = \Lambda$: Compatibility equations $\Leftrightarrow B\Lambda = 0$.

C : Diagonal matrix of elastic constants of the links.

$y = C\Lambda$: Hook's law (internal forces in the links).

f : External forces applied at the free nodes.

$A^T y = f$: Equilibrium equations (force balance at the nodes).

$K = A^T CA$: Stiffness matrix of a structure,

So far the compatibility equations looked like $B\Lambda = 0$, where B is a matrix expression that gives conditions on elongations (the compatibility matrix [11]); for example,

$$B = [1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; -1 \;\; -1 \;\; -1 \;\; -1 \;\; -1 \;\; -1],$$

while

$$\Lambda = [\lambda_{1,0} \;\; \lambda_{2,0} \;\; \lambda_{3,0} \;\; \lambda_{4,0} \;\; \lambda_{5,0} \;\; \lambda_{6,0} \;\; \lambda_{1,2} \;\; \lambda_{2,3} \;\; \lambda_{3,4} \;\; \lambda_{4,5} \;\; \lambda_{5,6} \;\; \lambda_{6,1}]^T$$

in a regular hexagon problem.

However, in the experiments that follow, the equivalent formulation for the compatibility equation is used. Namely, the problem that is being solved, using the Maple algebra system, is

$$A^T C\Lambda = f, \text{ subject to } \Lambda = AU. \tag{9.1}$$

The solutions to this linear system are given in the nodes' instantaneous velocities, and do not represent the actual displacements, because those are considered to be small enough to be linearized. The 2-D structures used in the experiments are the triangulated parallelogram-like grids (Figure 3.20 on page 34 and Figure 3.21 on page 35).

When writing the graphing program, the vertical displacements of horizontal links are allowed to *see* the node displacements, because the deformation is assumed to be small. One therefore has to keep in mind

that the diagrams or the graphs in this chapter represent the exaggerated deformations of structures.

In addition, what may seem like a very elongated horizontal link on the graph (this happens when the vertical displacement of a horizontal link on the graph looks to be big), in reality may not be nearly as elongated, because horizontal links do not get displaced vertically in a linearized approximation.

One of the biggest challenges when writing a program is simplification of indices. The next subsection deals with it.

9.2 The Numbering of Nodes and Links

Let $Q_\tau^2(S_{r\times c})$ be a triangulation with r row-nodes and c column-nodes. The structure given in Figure 3.20 on page 34 is $Q_\tau^2(S_{4\times 12})$, because it is a 4×12 grid. The counting of nodes is simple: it goes row by row. In other words,

$$n_{i,j} = a_{(i-1)\cdot c+j}, \text{ for all } i = 1, \ldots, r \text{ and } j = 1, \ldots, c,$$

where $n_{i,j}$ is a node in the i-th row and the j-th column.

The links are numbered as shown in Figure 3.21 on page 35, for any $Q_\tau^2(S_{r\times c})$.

We start at a_1 in the following manner: the first link is $l_1 = l_{1,2}$, or the link directly "West" to a_1, and then we move around a_1 in a clockwise direction: West (W) $\rightarrow$ Southwest (SW) $\rightarrow$ Southeast (SE) $\rightarrow$ East (E) $\rightarrow$ Northeast (NE) $\rightarrow$ Northwest (NW). Therefore, $l_2 = l_{1,6}$ and $l_3 = l_{1,5}$, which is when we stop circling a_1 because there are no links in the other three directions. Then we move to a_2 and repeat the procedure. We do this for every node in an ascending order, as shown in Figure 3.21 on page 35.

9.3 The Experiments

What follows is several examples of triangular grids under various loadings, and their numerical analysis. First the studying of a cantilever under pressure is done. Its resilience to the loading is increased by strengthening

certain links and weakening others while keeping its mass constant. Many different options are examined until the optimal solution is found, using the Maple code in Appendix E on page 130. It is worth mentioning that these are not the exact solutions, but rather very close approximations to what must be the optimal solutions, because the results are obtained by continually experimenting using educated guesses in the code.

The damage propagation through the cantilever under the same loading that was given in the first experiment is examined next. The studying of the rapid loss of its compatibility conditions, and ultimately its rigidity, aided in the conclusion that it is not advisable for every link of the cantilever to be of the same strength, as it then tends to lose its rigidity sooner than necessary, which also demonstrates the importance of the first experiment.

Examining a large triangular grid with all of its first-column nodes fixed, and loading applied to various free nodes, is done in Section 9.3.2 on page 87. The vulnerability of such a structure near the fixed nodes is observed, because all the links next to the fixed nodes are lost first. It is another reminder as to how important strengthening, as well as weakening, of certain links is, if the goal is to increase the structure's resilience to applied force, or to stay conditionally rigid until all of the compatibility conditions are exhausted.

Lastly, the studying of the same triangular grid from the previous experiment, with only two nodes fixed and loading applied, is done in Section 9.3.3 on page 89. One link in the middle of the structure is critically damaged, and the stresses in its neighboring links are observed, while the stresses everywhere else, especially near the fixed nodes, are ignored, because the interest is only in localized damage. The damage propagation forms a damage cluster that propagates in the somewhat intuitive fashion throughout the structure, destroying the compatibility conditions of affected hexagons.

In these examples, the *conditional rigidity*, rather than *global rigidity*, plays the crucial role. Namely, several links lingering, which would make a

structure globally flexible, does not necessarily make a structure collapse under the applied loading. As long as the energy of the loading is finite, the structure is conditionally rigid.

The optimal improvements of structures in the following experiments were not found by using optimization algorithms, but rather by numerical experimentation.

9.3.1 Cantilever

Let all the first-column nodes of $Q^2_\tau(S_{4\times 12})$ — $a_{1,1}$, $a_{2,1}$, $a_{3,1}$ and $a_{4,1}$ (using one-index notation that was used in the Maple code, the nodes are a_1, a_{13}, a_{25}, and a_{37}) — be fixed, and let the loading be applied to a_{48} (or $a_{4,12}$) directly downwards, which is the last node (Figure 9.3 on page 92).

As mentioned before, the deformation is small enough to be linearized. So, in order to make sense of the differences in displacements of various nodes, the velocity field of the structure in the linearized elongation equations is used as the node displacements. That is, the instantaneous velocities of the nodes are used in graphing the exaggerated version of its deformation.

Each of the 113 links is of the same length and is given the same spring constant (or stiffness), namely 1, which can be thought of as having each link made of the same material with the same mass. When the force is applied to the last node, a_{48}, directly downward, some links are under more stress than others.

The goal of this first experiment is to improve the cantilever's resistance to the applied force by strategically strengthening and weakening its links, while keeping the total mass constant. The improvement of a cantilever is measured by the magnitude of the vertical displacement of the last node, the one the force has been applied to — the smaller the vertical displacement is, the stronger the structure.

Strengthening a link is done by increasing its stiffness (spring constant), or, equivalently, by making it thicker, which is the same as

uniformly adding more mass to it. Similarly, a link is weakened if its stiffness, or, equivalently, its mass, is decreased. Initially, every link has the same length and the same stiffness (or mass), namely one, and because there are 113 links altogether, the mass of $Q^2_\tau(S_{4\times 12})$ is equal to 113, which is kept constant even after the improvements are made — the amount of material used and the design of the structure remain the same before and after the improvement.

The code given in Appendix E on page 130 looks for the links that are under the most stress — the most compressed and the most stretched links. The strengthening of a link is done manually in the code by increasing the stiffness constant to a desired amount of a wanted link. Once a link is chosen, it is strengthened by gradually increasing its stiffness, say by 0.1, until the maximum improvement is reached. That happens when the last node's displacement starts to increase after the next 0.1 is added. This implies that the structure becomes weaker as more mass gets added to it. Once this point is reached, the strengthening of this link stops, and the stiffness found is said to be optimal.

Because the mass of the cantilever has to stay constant, all the links that are not strengthened need to be weakened. This mass redistribution is automated in the code in the following way: first the link, or links, to be strengthened are picked, say l_k. Then the amount by which this link is going to be strengthened is chosen, say $\mu_1 = x$, where μ_i represents the amount added to the link's stiffness constant, and i is just an index distinguishing one μ from another. Then every single link is weakened, including l_k, by $\frac{x}{113}$; i.e., each link has the new mass (or stiffness) of $c_j - \frac{x}{113}$, where c_j represents the stiffness of l_j for all $j = 1, \ldots, m$, before this weakening step, and which is equal to 1 before any strengthening or weakening is done. Finally, the chosen amount is added to the mass of l_k, so that its new mass becomes $\left(c_k - \frac{x}{113}\right) + x$, where c_k *was* the mass or stiffness of l_k, the link picked to be strengthened. This is done for each link and each μ_i for all i.

9.3.1.1 Free Choice of Links

If the choice of the links to be strengthened or weakened is free, in an attempt to make the cantilever as resistant to the given applied force as possible, then this would, eventually, lead to an optimal stiffness- or mass-distribution of all links, which would result in the optimal cantilever's resistance to the applied force.

The following is the algorithm for constructing an approximation of the optimally resistant 4×12 cantilever that has the first-column nodes fixed, and a force applied to the last node, a_{48}, directly downward — this situation can be seen in Figure 9.1 on page 91. The weakest links are identified by their stresses: they are the most stretched and the most compressed links. Then they are improved by small increments until the improvement stops, which is signified by, either, no significant upward improvement in the displacement of the last node, or, worse, the last node's displacement is starting to increase after administering an increment to the chosen link or links. This means that strengthening these links and weakening others is starting to make the structure weaker.

The first run of the code given in Appendix E on page 130 records the vertical displacement of a_{48} as

$$m_0 = -4.981327134,$$

where m_i represents the vertical displacement of a_{48} after the i-th step of improvements. This first run, where the cantilever's each link is of the same length and stiffness, and is pulled directly downward with force $= -0.1$ by the last node, a_{48}, is demonstrated in Figure 9.3 on page 92.

Preliminary experiments, where strengthening two links at a time (the most stretched and the most compressed) is done, show that the links closest to the fixed nodes in rows one and four are the most stressed, with the row-one links being the most stretched, and the row-four links the most compressed, which is why the first improvement step is a bigger jump than the rest: the first six links in rows one and four, namely links $l_1, l_4, l_7, l_{10}, l_{13},$

and l_{16} in row one, and $l_{103}, l_{104}, l_{105}, l_{106}, l_{107}$, and l_{108} in row four; they are strengthened until a_{48} reaches its optimal height, which is at $\mu_1 = 2.9$. The improvement this first step makes is undeniable:

$$m_1 = -2.886241020,$$

which makes the structure

$$\frac{-4.981327134}{-2.886241020} = 1.76 \text{ times stronger.}$$

Each next step checks for the most stretched and the most compressed link, and makes the best improvement on them. That is, each m_i, for $i > 1$, improves two links at time. The second improvement step made additional improvements on the first link in row 1, l_1, and the first link in row 4, l_{103}, to obtain

$$m_2 = -2.498360981.$$

The third made improvements on l_{19}, l_{100}, and l_{110}, which resulted in

$$m_3 = 2.414968056.$$

The improvements continued with each step, and it finally reached close to its optimal improvement at

$$m_9 = -2.071027550,$$

which makes the structure 2.41 times stronger from its initial configuration. The improvements slow down significantly after m_9.

By looking at Figure 9.2 on page 91 one can see the leveling off of the vertical displacements for a_{48}. In fact, further link-strengthening keeps bringing us closer and closer to an apparent asymptote, which has to exist because the mass of a cantilever is constant and finite, the improvements are monotonically increasing and are bounded by zero. Experiments convinced us that the asymptote is at about -2.0. The cantilever that has been strengthened to this point resists the applied force about 2.5 times better — it is 150% stronger than before any strengthening took place.

9.3.1.2 Strengthening One Row of Links

If only one row of links can be improved, at the expense of the rest, because the total mass of the cantilever is always the same, which row should be chosen in order to optimize the cantilever's resistance to the given applied force? Judging by the previous section, the best candidates appear to be the first or the last row.

And, indeed, these are the correct choices, because strengthening row three by any amount only weakens the cantilever. In fact, reducing row three by the optimal margin of -0.5 and distributing the mass to other links actually strengthens the structure slightly (by only 0.6%, though), while the reduction by the optimal 1.0 of row-two link masses (stiffnesses), and distribution of this mass to other links, strengthens the cantilever by almost 6%. Therefore, strengthening the row-two and/or the row-three links at the expense of the other two rows of links actually weakens the structure, while weakening them strengthens the structure slightly. So row-two or row-three links would be an unwise choice.

In contrast, strengthening the links of row one by the optimal $\mu_1 = 1.8$, reduces the displacement of a_{48} from -4.981327134 to -4.157255625, making the cantilever 1.1982 times, or 19.82% stronger, while increasing the stiffness of the last row by the optimal $\mu_1 = 3.4$ reduces the displacement of the last node to -3.752472765, making the cantilever 32.75% stronger. So the winner is row four. These are represented graphically in Figure 9.5 on page 93 and Figure 9.6 on page 93.

9.3.1.3 Strengthening Two Rows of Links

Unsurprisingly, if any two rows of links can be improved, the best choice is the links of rows one and four. However, their optimal stiffness may be somewhat surprising.

If the optimal $\mu_1 = 1.8$ is added to the stiffness of the links of row one, and the optimal $\mu_2 = 2.4$ is added to the stiffness of row four links, the cantilever is then strengthened by 86.64%, while stiffening both rows by an optimal

$\mu_3 = 2.7$, counterintuitively, makes a bigger difference: it improves the cantilever's resistance to the applied force so that the last node's displacement is up to -2.611358087, which implies that the structure is 90.76% stronger than the unstrengthened one. This means that strengthening both rows of links by one, optimal μ – it being $\mu_3 = 2.7$ in this case — makes the cantilever stronger than strengthening each by a μ that is optimal to both individually.

In addition, it is found that strengthening both rows individually by their optimal μ — one for row one, the other for row four — improves the cantilever only slightly more than strengthening both rows by the same, optimal μ. Namely, the last node's displacement is up to -2.596432014, which makes the structure 91.85% if each row is strengthened by its own optimal μ (Figure 9.8 on page 94), compared to 90.76% when μ was one and the same for both rows (Figure 9.7 on page 94). But all of these are still dwarfed by the optimal strengthening, which was explained earlier, making the cantilever stronger by 150%. These improvements are graphed in Figure 9.9 on page 95.

Having freedom to strengthen each link individually yields, unsurprisingly, the best results: the cantilever can be made 150% stronger, judging by the difference in the last node's displacement to which the force has been applied directly downwards. But this would require that each of the 113 links be of different stiffness (or different mass), which may be too costly and/or time consuming.

If only one row of links is allowed to be strengthened, then row four would be the best option, row one comes in second, while strengthening rows two and three actually makes the cantilever weaker, which is not too surprising, because these rows of links are strengthened at the expense of rows one and four. But weakening these two rows of links weakens the cantilever.

Having the freedom to strengthen any two rows of links yields the most surprising results: while choosing two different stiffnesses for rows one and four does strengthen the cantilever the most, it is almost negligibly

stronger than the cantilever in which only one optimal stiffness is used for both rows. Therefore, if time and cost are issues, having just two kinds of links — one of stiffness $l_a = 3.174$ and the other of stiffness $l_b = 0.4743$, and having rows one and four be links of stiffness l_a while all the rest are the links of stiffness l_b — would make the cantilever almost as strong as if links of three different stiffnesses were used.

This cantilever, or $Q^2_\tau(S_{4\times 12})$, with four nodes fixed (which is eight parameters fixed), resulting in three dead links, has 20 inner nodes, and therefore $20 + 8 - 3 - 3 = 22$ compatibility conditions, not including the dead links. Another way of computing the number of compatibility conditions is to use the overused formula of this document, namely (3.2), where $C_d = 113 - 2 \cdot 48 + 8 = 25$, but this includes the three dead links. Regardless, what this says is that it can lose a maximum of twenty-two links, not including the dead ones, and still stay rigid. Strengthening it reduces the chance of losing compatibility conditions, hence improves its rigidity's longevity.

Stiffer structure does not necessarily mean that it can withhold more punishment. In fact, it is quite possible that, after the initial breakage, such structure loses rigidity rapidly, even if the force is very small compared to the initial force that is needed for the breakage of the first link. In what follows, however, the assumption is that the force is constant and unchanging, thus the first breakage would necessarily imply the pending destruction of the structure, because it now has fewer links that can absorb the applied force.

9.3.1.4 Cantilever with Damaged Links

Suppose the situation is the same as before — a 4×12 cantilever with all of its links having mass and length equal to one, having all of its first-column nodes fixed, and with a negative vertical force applied to a_{48} — but this time the links can be critically damaged. More precisely, after each run of the code given in Appendix F on page 139, the most stretched link is

found, and it is made 1000 times weaker than the rest: its mass (or stiffness) is set to 0.001, mimicking a critically damaged link.

The damaged links behave almost as if they were broken, because their stiffness is made 1000 times smaller, so the experiment is very helpful in understanding the spread of damage and the loss of compatibility conditions, which ultimately result in the loss of rigidity. This can be seen in Figures 9.10 on page 95 to Figure 9.15 on page 98, where in Figure 9.10 on page 95 no link has been damaged yet, while in Figure 9.15 on page 98 five of them are critically damaged, and, if they were broken, the cantilever would be flexible. The critically damaged links are denoted by dotted lines in the figures.

Examining the color code in Figure 9.10 on page 95, one notices that the most fragile links are those next to the fixed nodes, l_1 in particular. The color code is as follows: the greener the link is, the closer to its undeformed length it is; the bluer it is, the more stretched the link is; and the redder it is, the more compressed the link is. In the previous subsection it was determined that this link needed the most strengthening of all stretched links, and it is the first one to be critically damaged (Figure 9.11 on page 96).

These figures are exaggerations, especially the later ones where the deformation is big, so the colors are not to be trusted completely. The stresses are computed using the code given in Appendix G on page 152, while the graphs are done using Appendix F on page 139, and the latter does not take into consideration the linearization of displacements.

In fact, all the distances between the nodes in these diagrams are given via the Pythagorean theorem, because, otherwise, the structure would not be compatible; that is, the nodes would not be connected. This makes visualizing the deformation geometrically possible, even if a little exaggerated.

Once l_1 is critically damaged, it relieves of stress those links next to it, and overloads the ones directly below and parallel to it. And indeed, the next link to be critically damaged is l_{38}, which is one such link, which is demonstrated by dotting it in Figure 9.12 on page 96. Interestingly enough,

if the damage was not as critical as is given in this problem, say if it made l_1 only three times weaker instead of 1000, the link next to it, l_4, would not be relaxed enough, and would be the next one to go. The results from the last subsection confirm this.

The third link to be critically damaged is the one just below l_{38}, namely l_{39}, which is the first diagonal link to be critically damaged, which should not be surprising, either, because the stress on that link is significantly increased as soon as l_{38} is damaged.

The fourth link that suffers critical damage is not the one that looks the bluest in the last diagram, l_6, rather, it is another parallel link just below the two previous critically damaged links, namely l_{72}. This is an excellent example of the role that linearization plays in computing versus graphing: the most stretched link has been computed to be l_{72} — where linearization *is* used — even though in the diagram it looks to be l_6. This, again, is to be expected given that the links just above it absorb much less energy after l_{39} is damaged, which, in turn, overloads l_{72}.

After l_{72} goes, the cantilever becomes flexible, and it would look far more bent had those links been broken, rather than critically damaged. However, when the next link breaks, the cantilever collapses completely, which is demonstrated in Figure 9.15 on page 98

It needs to be emphasized that the cantilever's deformations, shown in Figures 9.10 on page 95 to Figure 9.15 on page 98, are exaggerations, especially the later figures. The apparent widening of the links to the right of the crack, for example, is a consequence of the linearization of the displacements. Nevertheless, this experiment clearly shows that the most vulnerable links are the ones in the first row, if the links can be critically damaged only by overstretching, and not by overcompressing, and those are the ones in need of strengthening, if one wants to avoid the quick loss of rigidity. An even better strategy is to strengthen the last row of links, as can be seen in the last subsection, which would relieve of stress the first row of links.

9.3.2 Loss of Compatibility Conditions Near Fixed Nodes

Let a 10×14 triangular grid be squeezed by forces applied to the nodes on the top row, and the nodes on the bottom row: Figure 9.16 on page 98. How does damage propagate, using the same assumption as before, that the links that are the most stretched get damaged?

The loading in the upper row is applied as follows: the force on a_4 is equal to -0.01, which means it is directly downwards, and it increases by -0.001 with each node to the right until it reaches a_{13}, which is the last node in the first row the force is applied to.

The loading in the last row is applied as follows: The force on node a_{131} is equal to 0.01, which means it is directly upward, and it increases by 0.003 with each node until the last node, a_{140}, is reached.

Most likely because of the small differences in strengths of the forces in the first row versus the forces in the last row, a big difference in the link stresses is observed, especially the first-column links. Because there are 373 links and $14 \cdot 10 - 2 \cdot 10 = 260$ degrees of freedom, this structure has $373 - 260 = 113$ compatibility conditions, where the nine trivial ones are counted, as well — the nine dead links in between the fixed nodes. Therefore, this structure can lose a maximum of 104 links and still stay globally rigid. It is possible for it to lose more than 104 links and stay conditionally rigid; that is, the structure may be globally flexible in some part that is not as crucial in keeping its conditional rigidity as some other parts, that are still globally rigid, are. But this task is not the topic of this document.

However, the number of compatibility conditions in the first column of links, that is, those links that are connected to the fixed nodes, is $10 - 1 = 9$. Thus, this structure can become flexible if the ten first-column links are gone — breaking nine of them would leave the structure compatibility-conditionless, but not flexible.

Interestingly enough, the first link to be critically damaged is not the first-column link, but the link just next to the first first-column link, namely

l_{362} (Figure 9.17 on page 99). The first first-column link was the second most stretched link, but after the link next to it gets critically damaged, the pressure on the first first-column link is relaxed, and it is transferred to the two links above and parallel to the damaged one. And, indeed, the two most stretched links are now those two, namely the second first-column link, l_{321}, and the one next to it, l_{324}. Unsurprisingly, the next link to be critically damaged is the second first-column link (Figure 9.18 on page 99).

The same happens after this link is critically damaged — the one next to it is relaxed, and the energy is absorbed by the two links just above and parallel to it. Again, the one connected to the fixed node loses, thus gets critically damaged (Figure 9.19 on page 100), while the one next to it is now relieved of pressure, and the trend continues until the sixth first-column link, l_{121}, is damaged (Figure 9.20 on page 100).

At this point, enough pressure has been collected by the forgotten first first-column link, l_{361}, that it finally succumbs to it, and becomes the next critically damaged link. After this the trend continues, and the next critically damaged link is l_{81} (Figure 9.21 on page 101).

All but two first-column links are damaged, and losing one more would make this structure flexible. However, because these links were not broken but only critically damaged, the next first-column damaged link would not make this structure collapse, as can be seen in Figure 9.22 on page 101. However, the next critically damaged link makes the structure collapse: see Figure 9.23 on page 102.

Even though the configuration of damaged links depends highly on the loading, the common trend seems to be that the most vulnerable links are those connected to the fixed nodes, almost independently of the loading.

In the next example the triangulated grid is assumed to be infinitely big, so all but the local links close to the artificially damaged link are studied. The purpose of this experiment is to track how the damage propagates from the middle outward.

9.3.3 Propagation of Damage

One of the shortcomings of the code used for studying the damage of structures in this document is that at least three parameters have to be fixed, and the links closest to them are always the most vulnerable, almost independently of the loading. But what if one is interested in the way the damage propagates away from fixed nodes? For this reason an experiment is conducted where one of the links in the middle of the grid is damaged, and the stresses of only those links around it are recorded.

The experiment is designed on a 10×14 triangular grid, again, with only two fixed nodes, namely a_1 and a_{14}, and with the forces applied as shown in Figure 9.24 on page 102, that causes the structure to deform as shown in Figure 9.25 on page 103. The link that is artificially critically damaged is l_{262} (Figure 9.26 on page 103). This single damaged link affects four hexagons that share it; and, because a hexagon is the smallest unit of compatibility of this grid — it is the smallest structure in this particular triangulated grid, which has at least one compatibility condition — each of these four hexagons loses its compatibility condition when this link is critically damaged. The border of the four affected hexagons is shown by an increased width of the links in Figure 9.27 on page 104; two of the four hexagons have this damaged link as one of their radial links, while for the other two it is one of the circumferential links.

In an undamaged structure, every single hexagon has one compatibility condition, and each of these is represented by different equations — they are all in the form of the wagon wheel condition. However, once the link gets critically damaged, the locality of this link gets affected. In fact, the four hexagons that share this link lose their compatibility condition — the hexagons that get affected by this single critically damaged link are clearly visible in the Figure 9.27 on page 104.

Each of these hexagons are different, which gives rise to different compatibility conditions; but these compatibility conditions are not independent, because the intersection of the four links is not empty. That is why,

even though four hexagons are affected, only one compatibility condition is lost, *globally*.

The same phenomenon that was observed in the previous experiments occurred here, as well. Namely, after this first link was critically damaged, the links in the same row, especially those next to it, were relaxed, while those directly above it and below it and parallel to it became more stressed. Unsurprisingly, one of those links was the next most stretched link in the neighborhood of l_{262}, and it is the next critically damaged link that affects two additional hexagons; so the affected area, that we call *the cluster of damage*, becomes that much bigger (Figure 9.28 on page 104).

With each critically damaged link the cluster of damage becomes bigger (Figures 9.29 on page 105 to Figure 9.31 on page 106), and the propagation of damage continues until the cluster of damage touches down on both ends, at which point the structure loses all compatibility conditions along this strip, resulting in the loss of conditional rigidity.

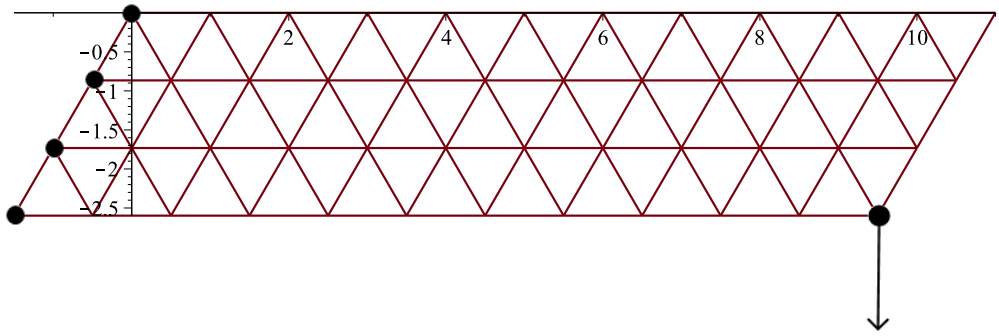

Figure 9.1: The 4×12 cantilever with the first column of links fixed and the force applied to the last node, n_{48}.

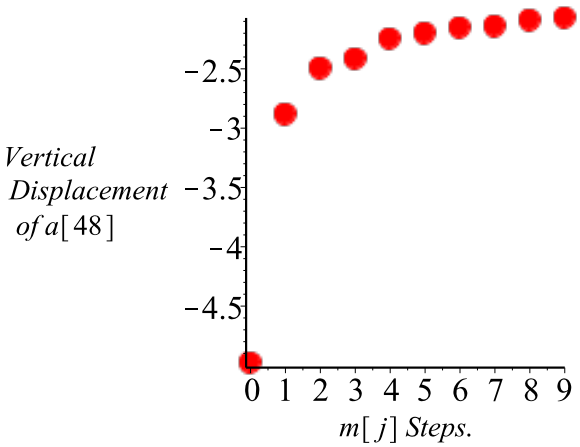

Figure 9.2: The improvement of the last node's displacement. m_9 is very close to the optimal improvement.

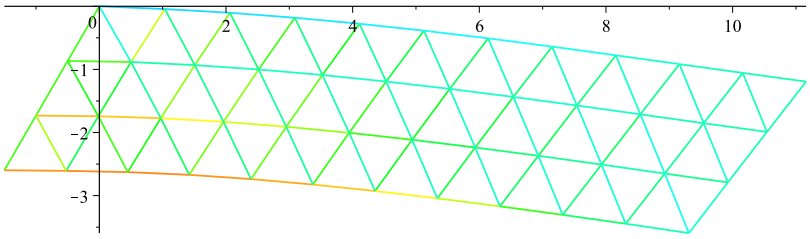

Figure 9.3: This is the original, unimproved cantilever, with every link having the same stiffness of 1. The colors represent the stresses of links: the more red it is, the more compressed the link is, the more blue it is, the more stretched the link is. The closer the link's color is to the green of links $l_3, l_{37},$ and l_{71}, which are the fixed, stationary links, the less stressed it is.

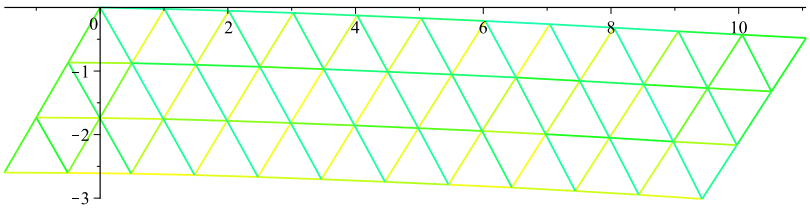

Figure 9.4: Last improvement.

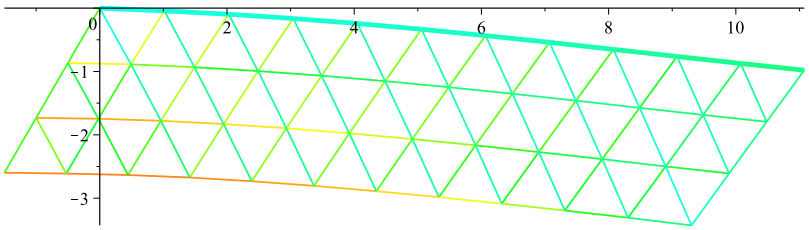

Figure 9.5: The first row of links is strengthened by the optimal $\mu_1 = 1.8$.

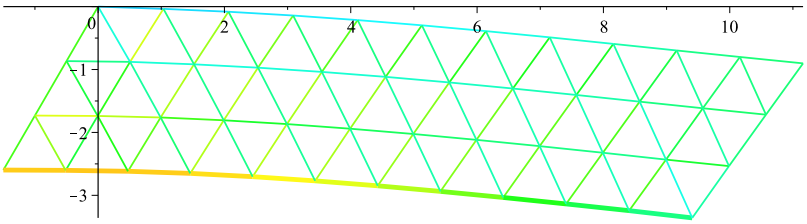

Figure 9.6: The last row of links is strengthened by the optimal $\mu_1 = 3.4$.

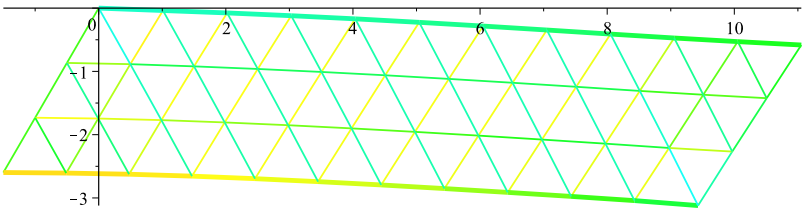

Figure 9.7: Here both the first and the last rows of links are strengthened by an optimal $\mu_3 = 2.7$.

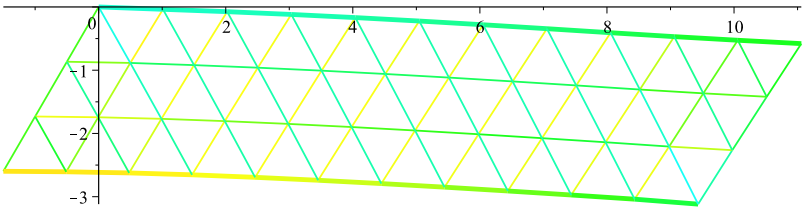

Figure 9.8: This is just a slight improvement over the previous figure: row 1 links are strengthened by $\mu_4 = 2.3$, while the last row is strengthened by $\mu_5 = 3.0$. Both together give the optimal improvement if only two rows of links are strengthened.

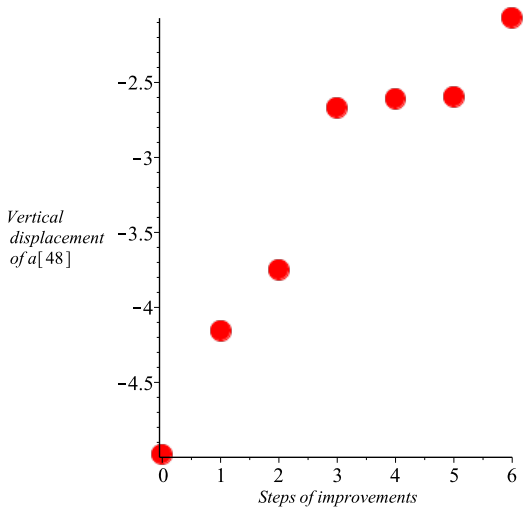

Figure 9.9: The first dot represents the original, unimproved cantilever's last node's displacement. The second is when the first-row links are strengthened to its optimal stiffness of 2.8. The third is when the last-row links are strengthened to its optimal stiffness of 3.4. The fourth is 1 and 2 combined, while the fifth is when the stiffness of both rows equals the optimal 3.7. The sixth dot represents the last node's displacement when the first and the last rows of links are strengthened to their combined optimal: 2.3 for row one, and 4.0 for row four. The last dot represents the optimal improvement of the last node's displacement.

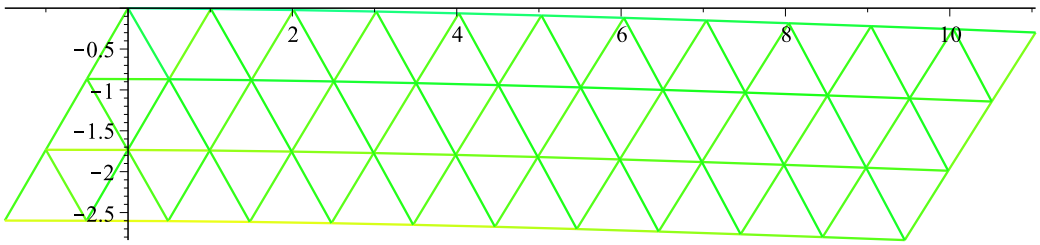

Figure 9.10: No link is damaged yet.

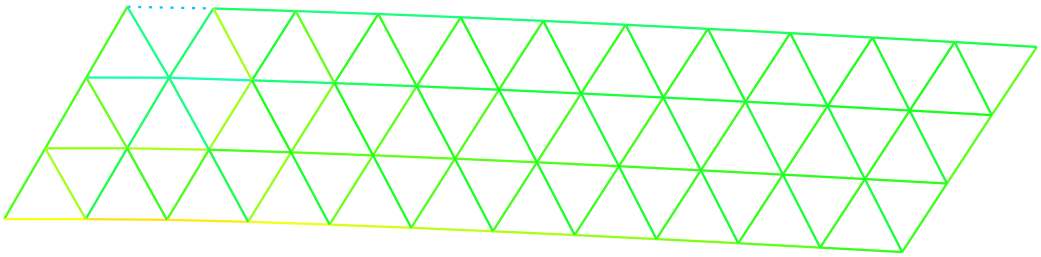

Figure 9.11: l_1 is the first damaged link.

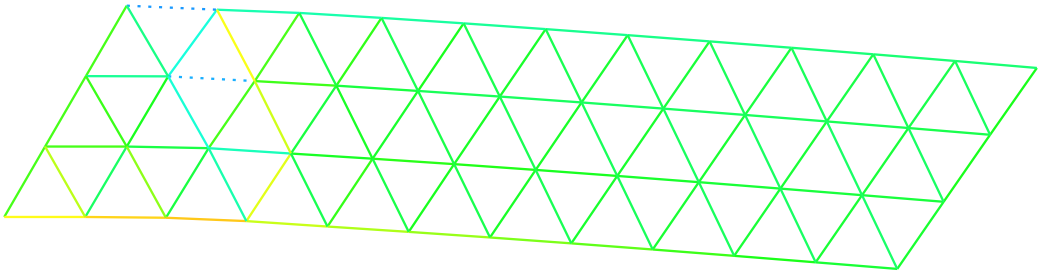

Figure 9.12: l_{38} is the second damaged link.

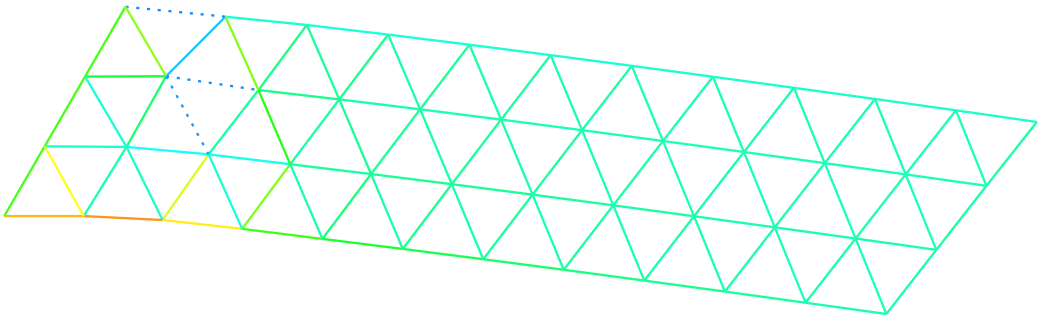

Figure 9.13: The third link that is badly damaged is l_{39}.

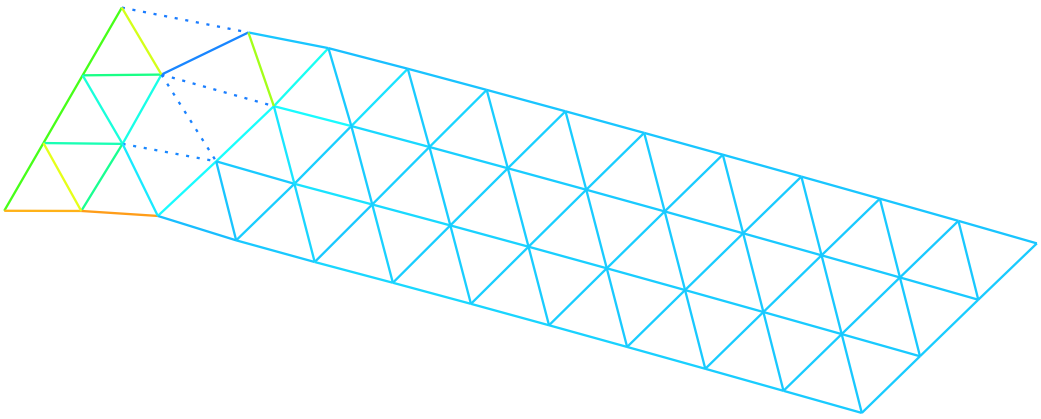

Figure 9.14: l_{72} is the fourth damaged link.

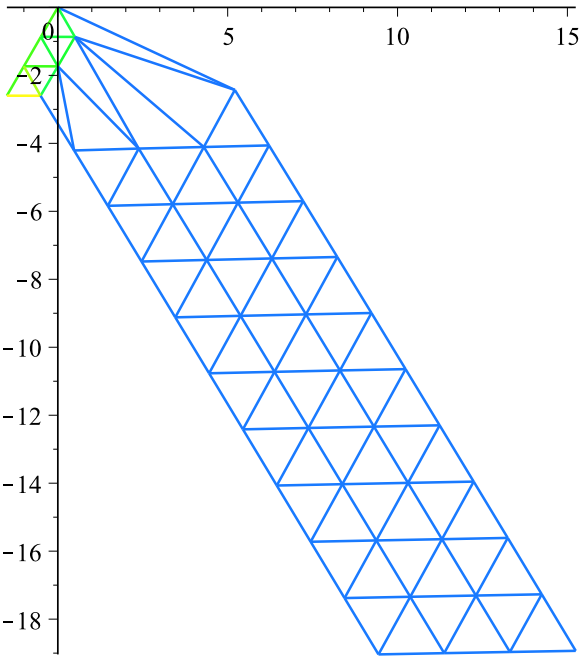

Figure 9.15: The next one to go is l_{73}, which causes the cantilever to collapse.

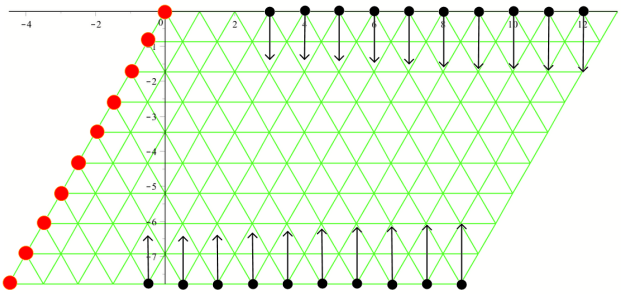

Figure 9.16: 10×14 triangular grid, where the green color of links indicates the grid is at rest. The first-column nodes are fixed, and the loading is applied to the first and the last rows of nodes.

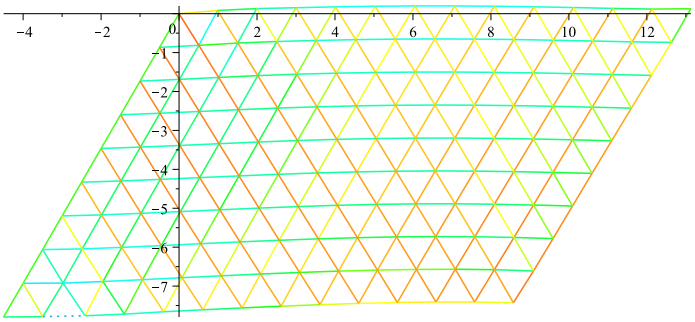

Figure 9.17: The first link to be critically damaged is l_{362}.

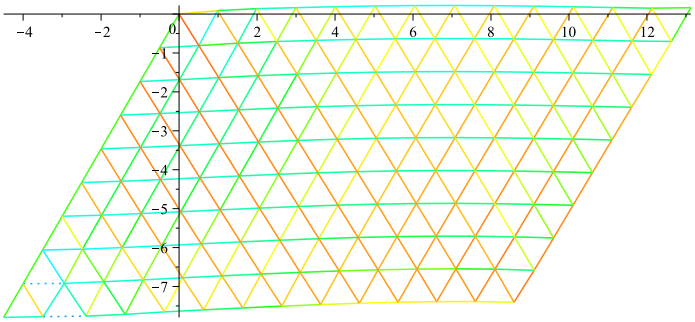

Figure 9.18: Second link to be critically damaged is l_{321}.

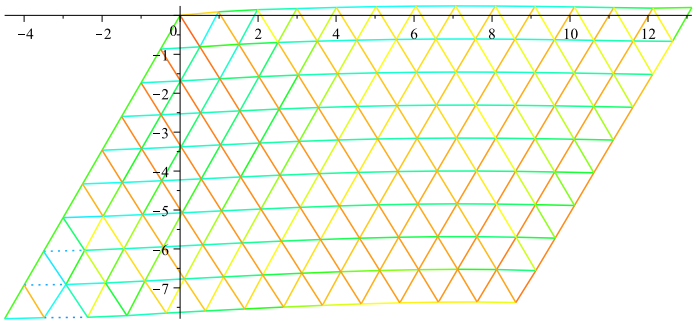

Figure 9.19: The trend continues, with the first-column links being damaged first: the third one to go is l_{281}.

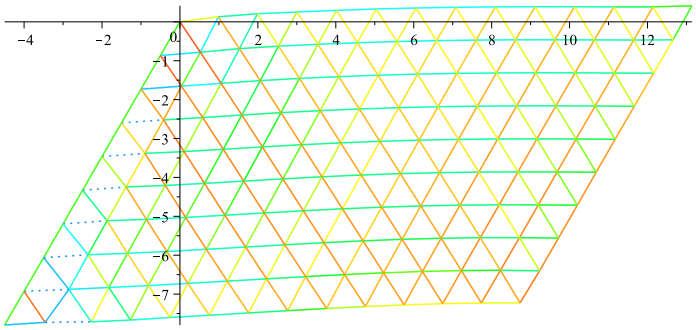

Figure 9.20: The damage of first-column links stops at l_{121}.

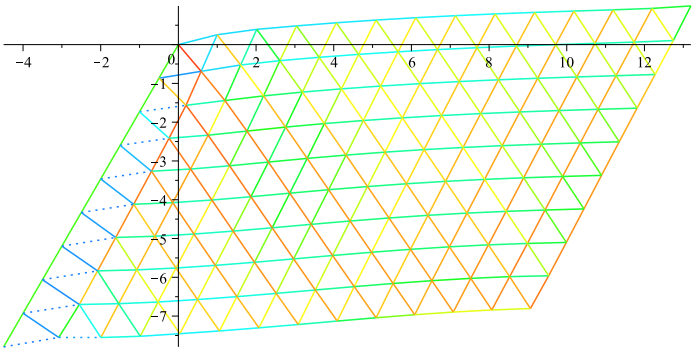

Figure 9.21: Finally, the first first-column link gets critically damaged.

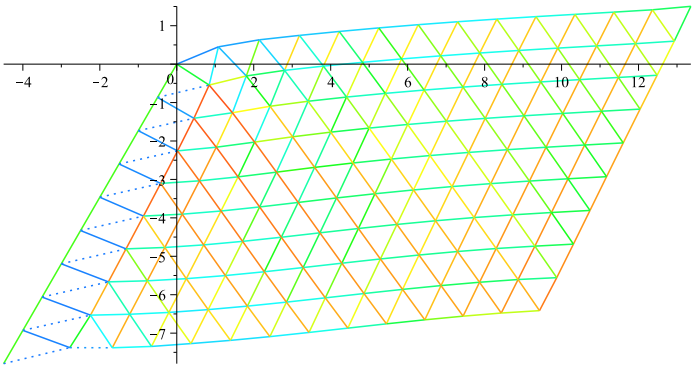

Figure 9.22: Hanging by a trend.

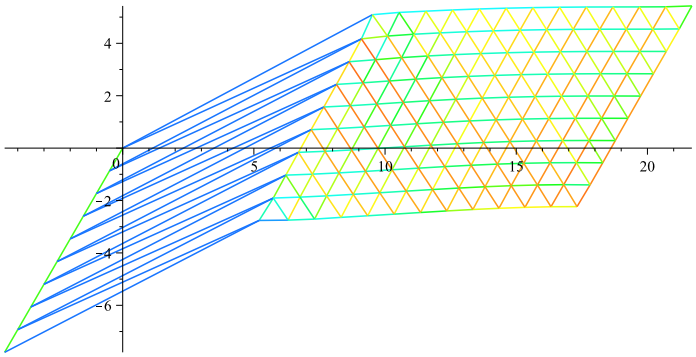

Figure 9.23: This is what it looks like when a structure loses rigidity.

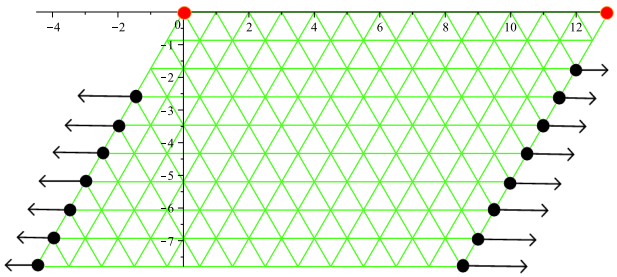

Figure 9.24: The two nodes at the two top corners are fixed, and the forces are applied to the first and the last column links, as shown in this picture.

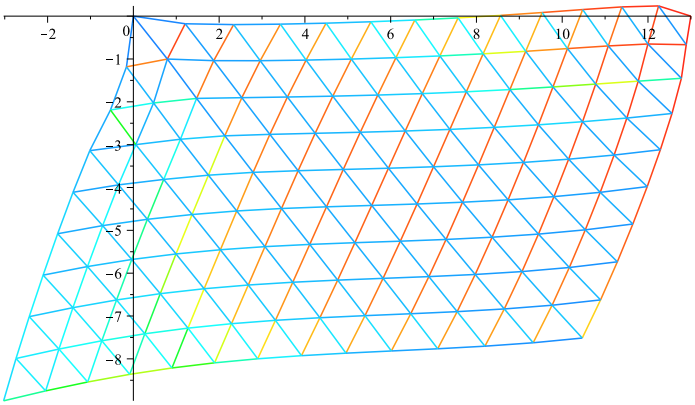

Figure 9.25: This is what the structure looks like when the forces are applied. No link is yet critically damaged.

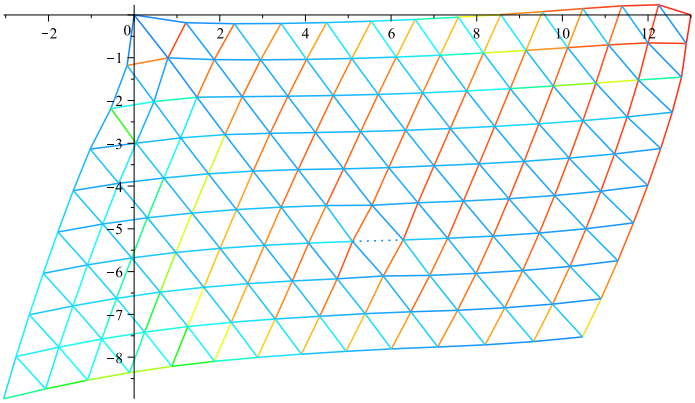

Figure 9.26: The first link is artificially critically damaged. It is the link somewhere in the middle of the structure, l_{262}, shown here as a dotted line.

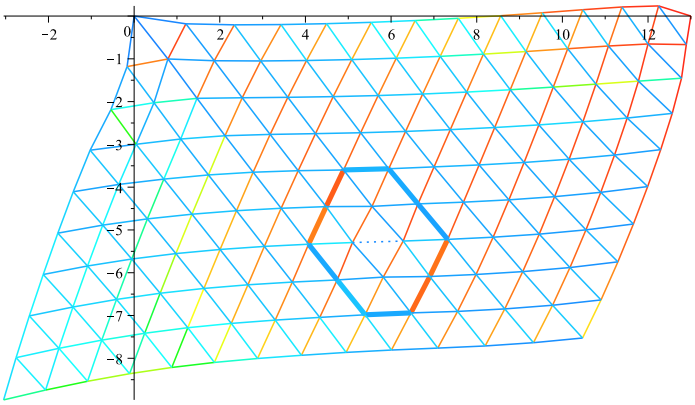

Figure 9.27: This is the area that gets affected by the damage of link l_{262}.

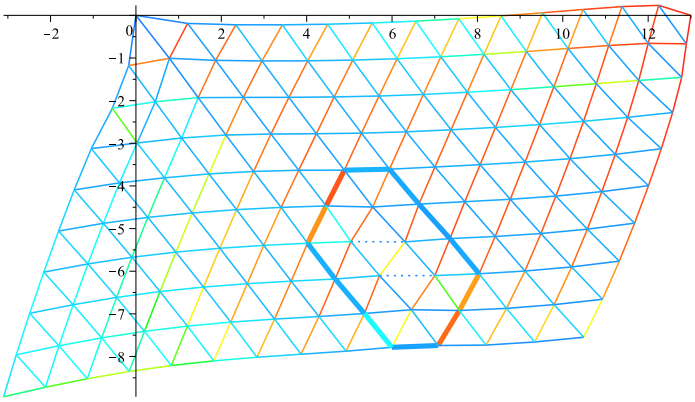

Figure 9.28: The number of affected hexagons increases with the number of critically damaged links.

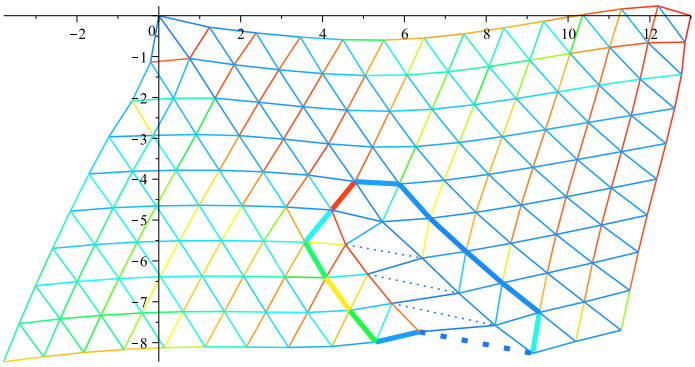

Figure 9.29: The spread of damage continues.

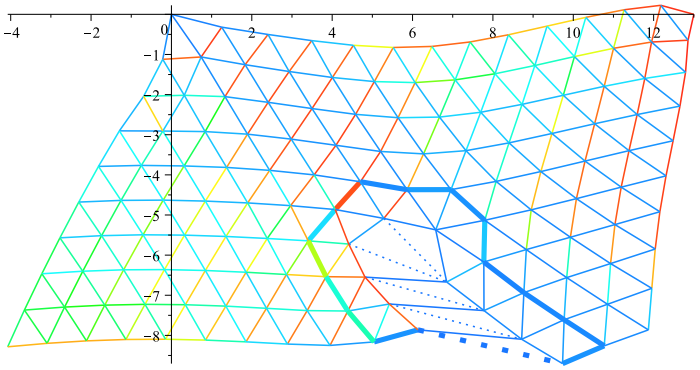

Figure 9.30: Another link damaged, another hexagon affected.

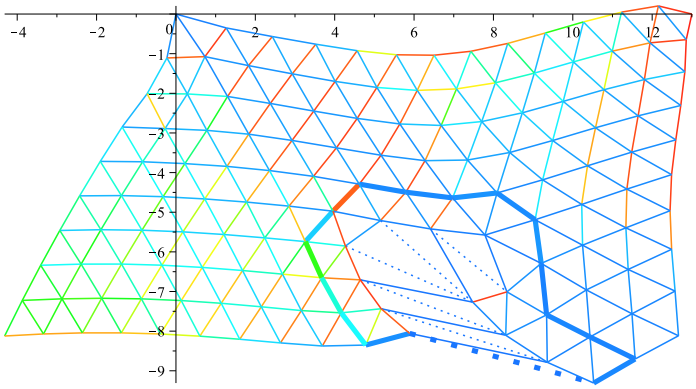

Figure 9.31: Even more hexagons affected.

CHAPTER 10

CONCLUSION

The following are the findings of this research.

1. Different shortcuts in counting discrete compatibility conditions of various structures were shown in Chapter 3 on page 8. The number of compatibility conditions, C_d, of discrete structures tells us the maximum number of links that can be broken before the structure loses its geometrical or global rigidity. Therefore, C_d represents a rough resilience scale of structures — the more compatibility conditions a structure has, the more stress-resilient it is, in general.

2. Conditional rigidity,[1] which depends on loading and fixed nodes, may be able to sustain more damage than its global C_d suggests, as it may lose those links that are not as critical in keeping the structure from collapsing under the applied force. In addition, each added fixed node increases the number of compatibility conditions, as is explained in Chapter 3 on page 8. Thus, some links are more important in keeping the structure's conditional, as well as global,[2] rigidity than others, which is why making a structure's critical links stronger at the expense of noncritical links is crucial in increasing its stress resilience. This is demonstrated by the cantilever experiments in Section 9.3.1 on page 78, where the importance of strategy of which links to strengthen

[1] If a linear system with prescribed support and loading has a linearized solution, then we call such a system conditionally rigid.

[2] Every 2-D triangulated structure can lose global rigidity after having only two links removed, even if its $C_d \gg 1$.

and which to weaken, while keeping the total mass constant, is made clear.

3. Connecting the wagon-wheel compatibility condition of a hexagon to a continuum 2-D compatibility condition is one of the biggest contributions made in this document. Namely, we show in Chapter 7 on page 61 that, under some reasonable assumptions about the strains of a continuum, and using the fact that hexagons centered anywhere in the continuum and of any size satisfy the wagon-wheel condition, we can deduce the continuum compatibility condition; thus connecting the discrete wagon-wheel compatibility condition with the continuum compatibility condition.

4. Computing compatibility conditions by the method of projections in Fourier space, in both, 2-D and 3-D, is done in Section 5.3 on page 46 by using the fact that the difference between the number of arguments and the number of components gives rise to a space of orthogonal projections. Equivalence between the differentiation and multiplication by the Fourier transform is also demonstrated, resulting in the reduction of finding compatibility conditions to finding orthogonal projections.

5. We consider close packing of equal spheres in Chapter 8 on page 68, leading to studying the smallest regular figure arising from close packing: cuboctahedron. Using a similar technique as in computing the compatibility conditions of hexagons, that we call the wagon-wheel conditions, we compute compatibility conditions of a cuboctahedron. Using any of the methods for counting compatibility conditions developed in Chapter 3 on page 8 we concluded that a regular cuboctahedron has three compatibility conditions. What we get, however, when we use linear algebra, is four compatibility conditions, each representing a wagon-wheel condition of one of the four intersecting hexagons any regular cuboctahedron has, and the extra condition is

attributed to its nongeneric nature. Namely, a regular cuboctahedron is infinitesimally flexible, so it allows for additional degrees of freedom.

However, if we perturb the nodes by any nonzero amount, we get a generic cuboctahedron that does have exactly three compatibility conditions — the breaking of the symmetry of a regular cuboctahedron makes it generic and moves away from the four planar compatibility conditions, and returns only three, as expected. This is an interesting demonstration of the role the infinitesimal rigidity plays in solving for compatibility conditions.

6. The importance of compatibility conditions in determining structure resilience, as well as the importance of strategic placing of stronger and weaker links within hexagonal structures to prolong their conditional rigidity, are demonstrated in the last chapter. It is made clear that controlling the spread of damage delays the conditional flexibility of structures.

   In addition, it is found that a critically damaged horizontal link relieves the pressure of the horizontal links to the left and right of it, while transferring the energy to the horizontal links directly above and below it, making those links the next likeliest candidates to be critically damaged.

APPENDIX A

SQUARE WITH DIAGONALS

```
restart;
with(PolynomialTools);
with(plots);
with(plottools);
with(OrthogonalSeries);
with(Student:-Precalculus);
with(ListTools);

#The following are the nodes coordinates:

a[0] := [0, 0]; a[1] := [-1/2, -1/2]; a[2] := [1/2, -1/2];
a[3] := [1/2, 1/2]; a[4] := [-1/2, 1/2];

#This routine computes the undeformed lengths of the links:

for j to 4 do L[0, j] := Distance(a[0], a[j]) end do;
L[1, 4] := Distance(a[1], a[4]);
L[1, 2] := Distance(a[1], a[2]);
L[2, 3] := Distance(a[2], a[3]);
L[3, 4] := Distance(a[3], a[4]);

#This graphs the structure:

plot1 := plot([a[3], a[1], a[2], a[4], a[3], a[2], a[4], a[1]],
thickness = 2, color = black); plot2 := plot([a[0], a[1], a[2],
a[3], a[4]], style = point, color = [black, black, red, black],
symbol = circle, symbolsize = 30);
display(plot1, plot2);

#The law of cosines is now used to compute the angles around
#the inner node in terms of the lengths and elongations:

for i to 3 do alpha[i] := arccos(((epsilon*lambda[i]+L[0,
i])^2+(epsilon*lambda[i+1]+L[0, i+1])^2-(epsilon*mu[i]+L[i,
i+1])^2)/((2*(epsilon*lambda[i]+L[0, i]))*(epsilon*lambda[i+1]+
L[0, i+1]))) end do;
```

```
alpha[4] := arccos(((epsilon*lambda[1]+L[0, 1])^2+
(epsilon*lambda[4]+L[0, 4])^2-(epsilon*mu[4]+L[1,
4])^2)/((2*(epsilon*lambda[1]+L[0, 1]))*(epsilon*lambda[4]+L[0, 4])));

#The Taylor series of the arccosines around epsilon = zero:

for i to 4 do s[i] := series(alpha[i], epsilon = 0, 2) end do;
for jj to 4 do ss[jj] := convert(s[jj], polynom) end do;
sm := sum(ss[jk], jk = 1 .. 4);

#The compatibility conditions:

sk := collect(simplify(sm-2*Pi), epsilon); simplify(sk/(2*epsilon));
```

APPENDIX B

AFFINE-REGULAR HEXAGON CODE

```
restart;
with(PolynomialTools);
with(plots);
with(plottools);
with(OrthogonalSeries);
with(Student:-Precalculus);
with(ListTools);

#TwoD = 0 is the 2-D continuum compatibility condition:
 TwoD := epsilon[11, 22]-2*epsilon[12, 12]+epsilon[22, 11];

#Below are the nodes of an affine-regular hexagon:
a[0] := [0, 0]; a[1] := [1, 0]; a[2] := [1-v, -w]; a[3] := [-v, -w];
a[4] := [-1, 0]; a[5] := [v-1, w]; a[6] := [v, w];
a[7] := a[1]; a[8] := a[2];

#This fixes the total angular momentum:
 f[1]:=(&sum;)(a[i][1]*u[i][2]-a[i][2]*u[i][1])=0;

#The l[i,j][k] represents the k-th coordinate of l_{i,j}.
 for ii from 0 to 6 do for jj from 0 to 6 do
 l[ii, jj][1] := a[ii][1]-a[jj][1] od od;

 for iii from 0 to 6 do for jjj from 0 to 6 do
```

```
 l[iii, jjj][2] := a[iii][2]-a[jjj][2] od od;

#The following computes the lengths of all 12 links:
 for ir from 1 to 6 do L[ir] := Distance(a[ir], a[0]) od;
 for ri from 1 to 5 do L[ri+6] := Distance(a[ri+1], a[ri]) od;
 L[12] := Distance(a[6], a[1]);

#Functions g[i]'s plus f, the function that fixes the total angular
 momentum, comprise the 13-equations-in-12-unknows system.
 for r from 1 to 6 do
   g[r] := l[r, 0][1]*u[r][1]+l[r, 0][2]*u[r][2] = L[r]*lambda[r, 0]
 od;

 for rr from 1 to 5 do
 g[rr+6] := (u[rr+1][1]-u[rr][1])*l[rr+1, rr][1]+
 (u[rr+1][2]-u[rr][2])*l[rr+1, rr][2] = L[rr+6]*lambda[rr+1, rr] od;

 g[12] := (u[6][1]-u[1][1])*l[6, 1][1]+
 (u[6][2]-u[1][2])*l[6, 1][2] = L[12]*lambda[6, 1];

#The next command solves the system of equations, and we get
#the relation between all lambda's, the link elongations.
 elim := eliminate({f[1], g[1], g[2], g[3], g[4], g[5], g[6], g[7],
 g[8], g[9], g[10], g[11], g[12]}, {u[1][1], u[1][2], u[2][1],
 u[2][2], u[3][1], u[3][2], u[4][1], u[4][2],
 u[5][1], u[5][2], u[6][1], u[6][2]});

#Picks the second list from elim. This is the wagon-wheel condition:
 cc:=elim[2][1];

#sp[i]'s represent the coefficients of spokes (or radial links),
```

```
#while ed[i]'s represent the coefficients of edges (or
#cirumferential links):
 for ii from 1 to 6 do sp[ii] := coeff(cc1, lambda[ii, 0]) od;
 for ii from 1 to 5 do ed[ii] := coeff(cc1, lambda[ii+1, ii]) od;
 ed[6] := coeff(cc1, lambda[6, 1]);

#Strains approximated by Taylor polynomials:
 M := proc (x, y) options operator, arrow;
 sum(sum(binomial(i, j)*m[i, j]*x^j*y^(i-j)/factorial(i),
 j = 0 .. i), i = 0 .. 3) end proc; simplify(M(x, y));

 N := proc (x, y) options operator, arrow;
 sum(sum(binomial(i, j)*n[i, j]*x^j*y^(i-j)/factorial(i),
 j = 0 .. i), i = 0 .. 3) end proc; simplify(N(x, y));

 P := proc (x, y) options operator, arrow;
 sum(sum(binomial(i, j)*p[i, j]*x^j*y^(i-j)/factorial(i),
 j = 0 .. i), i = 0 .. 3) end proc; simplify(P(x, y));

 epsilon[11] := M; epsilon[12] := N; epsilon[22] := P;

#Here starts the code for the Lemma.;
#lismul takes point V = (V[1], V[2]) and multiplies it by a
#scalar u:
 lismul := proc (V, u) options operator, arrow; [u*V[1], u*V[2]]
 end proc;

#Q is the integrand. In fact,
#Q = [[[T[1],T[2]]]]*[[[M(V[1],V[2]),N(V[1],V[2])],[N(V[1],V[2]),
 P(V[1],V[2])]]]*[[[T[1]],[T[2]]]]:
```

```
  Q := proc (V, T) options operator, arrow;
  M(op(V))*T[1]^2+2*N(op(V))*T[1]*T[2]+P(op(V))*T[2]^2 end proc

#lislc takes points V and W and multiplies by a and b:
 lislc := proc (V, W, a, b) options operator, arrow;
 [a*V[1]+b*W[1], a*V[2]+b*W[2]] end proc;

#This is the linear parametarization:
 c := proc (s) options operator, arrow;
 lislc(a[ll], a[ll+1], delta-s, s) end proc; c(t);

 omega := proc (s) options operator, arrow;
 lislc(a[ll+1], a[ll], 1, -1) end proc;
 omega(t); Q(c(t), omega(t));

#Computation of radial links (or spokes) elongations:
 for ll from 1 to 6 do
 sp1[ll] := collect(simplify(sum((int(Q(lismul(a[ij], s), a[ij]),
 s = 0 .. delta))/(delta*L[ll]), ij = ll .. ll)), delta) od;

#Computation of circumferential links (or edges) elongations:
 for ll to 6 do ed1[ll] := collect(simplify((int(Q(c(t), omega(t)),
 t = 0 .. delta))/(delta*L[ll+6])), delta) od;

#Multiplying each spoke elongation by its weight, summing them all
#up, and factoring out delta:
 sp2 := collect(simplify(sum(sp[jkl]*sp1[jkl], jkl = 1 .. 6)),
                delta);

#Multiplying each edge elongation by its weight, summing them all
#up, and factoring out delta:
```

```
 ed2 := collect(simplify(sum(ed[jki]*ed1[jki], jki = 1 .. 6)),
                delta);

#Now we apply the wagon-wheel condition to the integrated spokes and
#edges, and we check what the coefficient of delta^(2) is.
 cc2 := collect(ed2+sp2, delta); cc3 := coeff(cc2, delta^2);

 cc3 := -w^2*(epsilon[11, 22]-2*epsilon[12, 12]+epsilon[22, 11])

#Because the coefficient of delta^2 is a w^2 multiple of the LHS of
#the 2-D compatibility condition, the RHS of it being =0, we
#conclude that it holds!
```

APPENDIX C

WAGON-WHEEL CONDITION FOR REGULAR HEXAGON NOT CENTERED AT THE ORIGIN

```
restart;
with(PolynomialTools);
with(plots);
with(plottools);
with(OrthogonalSeries);
with(Student:-Precalculus);
with(ListTools);

#TwoD = 0 is the 2-D continuum compatibility condition:

 TwoD := epsilon[11, 22]-2*epsilon[12, 12]+epsilon[22, 11];

#(k, h) is the vector by which the regular hexagon centered at the
#origin is translated. I chose 1/2 and -1/3 here, but leaving it as
#a general k and h works too.

 k := 1/2; h := -1/3;

#Below are the nodes of a regular hexagon:

 a[0] := [0, 0]; a[1] := [1, 0]; a[2] := [1/2, (1/2)*sqrt(3)];
 a[3] := [-1/2, (1/2)*sqrt(3)]; a[4] := [-1, 0];
 a[5] := [-1/2, -(1/2)*sqrt(3)]; a[6] := [1/2, -(1/2)*sqrt(3)];
```

```
 a[7] := a[1]; a[8] := a[2];
 for ii from 0 to 8 do
   a[ii][1] := a[ii][1]+k; a[ii][2] := a[ii][2]+h
 od;

#Graphing the hexagon:

 plot1 := plot([a[0], a[1], a[2], a[3], a[4], a[5], a[6], a[1],
               a[2], a[0], a[3], a[4], a[0], a[5], a[6], a[0]],
               thickness = 2, color = black);
 plot2 := plot([a[0], a[1], a[2], a[3], a[4], a[5], a[6]],
               style = point,
               color = [red, black, black, black, black, black,
                        black],
 symbol = circle, symbolsize = 20); display(plot1, plot2);

#This fixes the total angular momentum:

 f[1]:=(&sum;)(a[i][1]*u[i][2]-a[i][2]*u[i][1])=0;

#The l[i,j][k] represents the k-th coordinate of l_{i,j}.

 for ii from 0 to 6 do for jj from 0 to 6 do
 l[ii, jj][1] := a[ii][1]-a[jj][1] od od;

 for iii from 0 to 6 do for jjj from 0 to 6 do
 l[iii, jjj][2] := a[iii][2]-a[jjj][2] od od;

#The following computes the lengths of all 12 links:

 for ir from 1 to 6 do L[ir] := Distance(a[ir], a[0]) od;
```

```
 for ri from 1 to 5 do L[ri+6] := Distance(a[ri+1], a[ri]) od;
 L[12] := Distance(a[6], a[1]);

#Functions g[i]'s plus f, the function that fixes the total angular
#momentum, comprise the 13-equations-in-12-unknows system.

 for r from 1 to 6 do
   g[r] := l[r, 0][1]*u[r][1]+l[r, 0][2]*u[r][2] = L[r]*lambda[r, 0]
 od;

 for rr from 1 to 5 do
 g[rr+6] := (u[rr+1][1]-u[rr][1])*l[rr+1, rr][1]+
 (u[rr+1][2]-u[rr][2])*l[rr+1, rr][2] = L[rr+6]*lambda[rr+1, rr] od;

 g[12] := (u[6][1]-u[1][1])*l[6, 1][1]+
 (u[6][2]-u[1][2])*l[6, 1][2] = L[12]*lambda[6, 1];

#The next command solves the system of equations, and we get
#the relation between all lambda's, the link elongations.

 elim := eliminate({f[1], g[1], g[2], g[3], g[4], g[5], g[6], g[7],
 g[8], g[9], g[10], g[11], g[12]}, {u[1][1], u[1][2], u[2][1],
 u[2][2], u[3][1], u[3][2], u[4][1], u[4][2],
 u[5][1], u[5][2], u[6][1], u[6][2]});

#Picks the second list from elim. This is the wagon-wheel condition:
 cc:=elim[2][1];

#sp[i]'s represent the coefficients of spokes (or radial links),
#while ed[i]'s represent the coefficients of edges (or
#cirumferential links):
```

```
 for ii from 1 to 6 do sp[ii] := coeff(cc1, lambda[ii, 0]) od;
 for ii from 1 to 5 do ed[ii] := coeff(cc1, lambda[ii+1, ii]) od;
 ed[6] := coeff(cc1, lambda[6, 1]);

#Strains approximated by Taylor polynomials:

 M := proc (x, y) options operator, arrow;
 sum(sum(binomial(i, j)*m[i, j]*x^j*y^(i-j)/factorial(i),
 j = 0 .. i), i = 0 .. 3) end proc; simplify(M(x, y));

 N := proc (x, y) options operator, arrow;
 sum(sum(binomial(i, j)*n[i, j]*x^j*y^(i-j)/factorial(i),
 j = 0 .. i), i = 0 .. 3) end proc; simplify(N(x, y));

 P := proc (x, y) options operator, arrow;
 sum(sum(binomial(i, j)*p[i, j]*x^j*y^(i-j)/factorial(i),
 j = 0 .. i), i = 0 .. 3) end proc; simplify(P(x, y));

 epsilon[11] := M; epsilon[12] := N; epsilon[22] := P;

#Here starts the code for the Lemma.;
#lismul takes point V = (V[1], V[2]) and multiplies it by a
#scalar u:

 lismul := proc (V, u) options operator, arrow; [u*V[1], u*V[2]]
 end proc;

#Q is the integrand. In fact,
#Q = [[[T[1],T[2]]]]*[[[M(V[1],V[2]),N(V[1],V[2])],[N(V[1],V[2]),
 P(V[1],V[2])]]]*[[[T[1]],[T[2]]]]:
```

```
  Q := proc (V, T) options operator, arrow;
  M(op(V))*T[1]^2+2*N(op(V))*T[1]*T[2]+P(op(V))*T[2]^2 end proc

#lislc takes points V and W and multiplies by a and b:

 lislc := proc (V, W, a, b) options operator, arrow;
 [a*V[1]+b*W[1], a*V[2]+b*W[2]] end proc;

#This is the linear parametarization of the spokes (radial links):

 v := proc (s) options operator, arrow; lislc(a[0], a[ll]-a[0],
              delta, s)
 end proc;

 w := proc (s) options operator, arrow; lislc(a[ll], a[0], 1, -1)
 end proc;

#And this is the linear parametarization of
#the edges (circumferential links):

 c := proc (s) options operator, arrow; lislc(a[ll], a[ll+1],
              delta-s, s)
 end proc;

 omega := proc (s) options operator, arrow; lislc(a[ll+1], a[ll],
                  1, -1)
 end proc;

#Computation of radial links (or spokes) elongations:
```

```
 for ll to 6 do sp1[ll] :=
 collect(simplify((int(Q(v(t), w(t)), t = 0 .. delta))/
         (delta*L[ll])),
  delta)
 od;

#Computation of circumferential links (or edges) elongations:

 for ll to 6 do ed1[ll] :=
   collect(simplify((int(Q(c(t), omega(t)), t=0..delta))/
           (delta*L[ll+6])), delta)
 end do;

#Multiplying each spoke elongation by its weight, summing them all
#up, and factoring out delta:

 sp2 := collect(simplify(sum(sp[jkl]*sp1[jkl], jkl = 1 .. 6)),
                delta);

#Multiplying each edge elongation by its weight, summing them all
#up, and factoring out delta:

 ed2 := collect(simplify(sum(ed[jki]*ed1[jki], jki = 1 .. 6)),
                delta);

#Now we apply the wagon-wheel condition to the integrated spokes and
#edges, and we check what the coefficient of delta^(2) is.

 cc2 := collect(ed2+sp2, delta); cc3 := coeff(cc2, delta^2);

 cc3 := -w^2*(epsilon[11, 22]-2*epsilon[12, 12]+epsilon[22, 11])
```

```
#Because the coefficient of delta^2 is a w^2 multiple of the LHS of
#the 2-D compatibility condition, the RHS of it being =0, we
#conclude that it holds!
```

APPENDIX D

EXAMPLE OF A GENERAL HEXAGON

```
#This code tests discrete-to-continuum theorem with
#an example of some general hexagon.

restart;
with(PolynomialTools);
with(plots);
with(plottools);
with(OrthogonalSeries);
with(Student:-Precalculus);
with(ListTools);

#TwoD = 0 is the 2-D continuum compatibility condition:
 TwoD := epsilon[11, 22]-2*epsilon[12, 12]+epsilon[22, 11];

#Below are the nodes of a hexagon, with v and w chosen so the
#hexagon is not affine-regular, and resembles any general hexagon:
 a[0] := [0, 0]; a[1] := [1, 0];
 a[2] := [1-v+10^(-1), -w+2*10^(-1)];
 a[3] := [-v, -w]; a[4] := [-1+3*10^(-2), 0+2*10^(-1)];
 a[5] := [v-1, w+3*10^(-1)]; a[6] := [v, w];
 a[7] := a[1]; a[8] := a[2]; v := 2*10^(-1); w := -2;

#The following graphs this hexagon:
 plot1 := plot([a[0], a[1], a[2], a[3], a[4], a[5], a[6], a[1],
```

```
                a[2], a[0], a[3], a[4], a[0], a[5], a[6], a[0]],
               thickness = 2, color = black);
 plot2 := plot([a[0], a[1], a[2], a[3], a[4], a[5], a[6]],
                style = point,
                color = [red, black, black, black, black, black,
                         black],
                symbol = circle, symbolsize = 20);
 display(plot1, plot2)

#This fixes the total angular momentum:
 f[1]:=(&sum;)(a[i][1]*u[i][2]-a[i][2]*u[i][1])=0;

#The l[i,j][k] represents the k-th coordinate of l_{i,j}.
 for ii from 0 to 6 do for jj from 0 to 6 do
 l[ii, jj][1] := a[ii][1]-a[jj][1] od od;

 for iii from 0 to 6 do for jjj from 0 to 6 do
 l[iii, jjj][2] := a[iii][2]-a[jjj][2] od od;

#The following computes the lengths of all 12 links:
 for ir from 1 to 6 do L[ir] := Distance(a[ir], a[0]) od;
 for ri from 1 to 5 do L[ri+6] := Distance(a[ri+1], a[ri]) od;
 L[12] := Distance(a[6], a[1]);

#Functions g[i]'s plus f, the function that fixes the total angular
 momentum, comprise the 13-equations-in-12-unknows system.
 for r from 1 to 6 do
   g[r] := l[r, 0][1]*u[r][1]+l[r, 0][2]*u[r][2] = L[r]*lambda[r, 0]
 od;

 for rr from 1 to 5 do
```

```
 g[rr+6] := (u[rr+1][1]-u[rr][1])*l[rr+1, rr][1]+
 (u[rr+1][2]-u[rr][2])*l[rr+1, rr][2] = L[rr+6]*lambda[rr+1, rr] od;

 g[12] := (u[6][1]-u[1][1])*l[6, 1][1]+
 (u[6][2]-u[1][2])*l[6, 1][2] = L[12]*lambda[6, 1];

#The next command solves the system of equations, and we get
#the relation between all lambda's, the link elongations.
 elim := eliminate({f[1], g[1], g[2], g[3], g[4], g[5], g[6], g[7],
 g[8], g[9], g[10], g[11], g[12]},
 {u[1][1], u[1][2], u[2][1], u[2][2],
  u[3][1], u[3][2], u[4][1], u[4][2],
  u[5][1], u[5][2], u[6][1], u[6][2]});

#Picks the second list from elim. This is the wagon-wheel condition:
 cc:=elim[2][1];

#sp[i]'s represent the coefficients of spokes (or radial links),
#while ed[i]'s represent the coefficients of edges (or
#cirumferential links):
 for ii from 1 to 6 do sp[ii] := coeff(cc1, lambda[ii, 0]) od;
 for ii from 1 to 5 do ed[ii] := coeff(cc1, lambda[ii+1, ii]) od;
 ed[6] := coeff(cc1, lambda[6, 1]);

#Strains approximated by Taylor polynomials:
 M := proc (x, y) options operator, arrow;
 sum(sum(binomial(i, j)*m[i, j]*x^j*y^(i-j)/factorial(i),
 j = 0 .. i), i = 0 .. 3) end proc; simplify(M(x, y));

 N := proc (x, y) options operator, arrow;
 sum(sum(binomial(i, j)*n[i, j]*x^j*y^(i-j)/factorial(i),
```

```
 j = 0 .. i), i = 0 .. 3) end proc; simplify(N(x, y));

 P := proc (x, y) options operator, arrow;
 sum(sum(binomial(i, j)*p[i, j]*x^j*y^(i-j)/factorial(i),
 j = 0 .. i), i = 0 .. 3) end proc; simplify(P(x, y));

 epsilon[11] := M; epsilon[12] := N; epsilon[22] := P;

#Here starts the code for the Lemma.;
#lismul takes point V = (V[1], V[2]) and multiplies it by a
#scalar u:
 lismul := proc (V, u) options operator, arrow; [u*V[1], u*V[2]]
         end proc;

#Q is the integrand. In fact,
#Q = [[[T[1],T[2]]]]*[[[M(V[1],V[2]),N(V[1],V[2])],[N(V[1],V[2]),
 P(V[1],V[2])]]]*[[[T[1]],[T[2]]]]:

  Q := proc (V, T) options operator, arrow;
  M(op(V))*T[1]^2+2*N(op(V))*T[1]*T[2]+P(op(V))*T[2]^2 end proc

#lislc takes points V and W and multiplies by a and b:
 lislc := proc (V, W, a, b) options operator, arrow;
 [a*V[1]+b*W[1], a*V[2]+b*W[2]] end proc;

#This is the linear parametarization:
 c := proc (s) options operator, arrow;
 lislc(a[ll], a[ll+1], delta-s, s) end proc; c(t);

 omega := proc (s) options operator, arrow;
 lislc(a[ll+1], a[ll], 1, -1) end proc;
```

```
 omega(t); Q(c(t), omega(t));

#Computation of radial links (or spokes) elongations:
 for ll from 1 to 6 do
 sp1[ll] := collect(simplify(sum((int(Q(lismul(a[ij], s), a[ij]),
 s = 0 .. delta))/(delta*L[ll]), ij = ll .. ll)), delta) od;

#Computation of circumferential links (or edges) elongations:
 for ll to 6 do ed1[ll] := collect(simplify((int(Q(c(t), omega(t)),
 t = 0 .. delta))/(delta*L[ll+6])), delta) od;

#Multiplying each spoke elongation by its weight, summing them all
#up, and factoring out delta:
 sp2 := collect(simplify(sum(sp[jkl]*sp1[jkl], jkl = 1 .. 6)),
                delta);

#Multiplying each edge elongation by its weight, summing them all
#up, and factoring out delta:
 ed2 := collect(simplify(sum(ed[jki]*ed1[jki], jki = 1 .. 6)),
                delta);

#Now we apply the wagon-wheel condition to the integrated spokes and
#edges, and we check what the coefficient of delta^(2) is.
 cc2 := collect(ed2+sp2, delta); cc3 := coeff(cc2, delta^2);
 cc4 := simplify(cc3/TwoD); pos := type(cc4, positive);
 neg := type(cc4, negative);

 cc3 := (1034483933637/25)*epsilon[11, 22]-
        (2068967867274/25)*epsilon[12, 12]+
        (1034483933637/25)*epsilon[22, 11]
```

```
#The next routine checks if the coefficient of delta^(2) is a
#constant multiple of TwoD.
 if or(pos, neg) then
   print('Compatibility*Condition*factorial(Holds)')
 else
   print('Compatibility*Condition*Does*NOT*factorial(Hold)')
 end if;
```

APPENDIX E

EXPERIMENT 1: THE CANTILEVER

```
 with(LinearAlgebra);
 r := 4;
 c := 12;

 printlevel := 2;

#The following routiness assign each node
#in row i and column j one index:

 for i from 1 to r do for j from 1 to c do
 n := proc (i, j) options operator, arrow; (i-1)*c+j end proc od od;

 for i from 1 to r do for j from 1 to c do
 print((i, j) = (i-1)*c+j) od od;

#The number of fixed parameters:

 p := 8;

#r is the number of rows of links, c is the number of columns.
#N is the total number of nodes, while L is the total number of
#links.

 N := c*r;
```

```
 L := (c-1)*r+(r-1)*c+2*(sum(kik, kik = 1 .. r-1))+(c-r-1)*(r-1);

#The number of degrees of freedom:

 M := 2*N-p;

#l[i,j] is the j-th link of the i-th node, where j is one of 6
#possible directions (E=1, SE=2, SW=3, W=4, NW=5 NE=6).
#There are no NW & NE links at the top (the fist column):

 for jj from 1 to c do for ii from 5 to 6 do
  l[n(1, jj), ii] := 0 od od;

#There are no W & NW links in the first column:

 for k from 1 to r do for kk from 4 to 5 do
  l[n(k, 1), kk] := 0 od od;

#There are no SE & SW links in the last row:

 for ji from 1 to c do for jij from 2 to 3 do
  l[n(r, ji), jij] := 0 od od;

#There are no E & SE links in the last column:

 for i from 1 to r do for j to 2 do l[n(i, c), j] := 0 od od;

#I define these to be :=0 just so I can use the next nested loop.
 for i from 1 to 6 do for j from 1 to 6 do l[n(0,i),j]:=0 od od;

#These are the W & SW links of the last column,
```

```
#preparing us for the big loop:

 for k from 1 to r-1 do for kk from 3 to 4 do
 l[n(k,c),kk]:=k*((c-1)*3+1)-3*(kk-3) od od;

#This big nested loop takes care of E, SE, and SW links:

 for ii from 1 to r-1 do for i from 1 to c-1 do
 for j from 1 to 3 do l[n(ii,i),j]:=j+l[n(ii,i-1),3] od od od;

#The last row links:

 for i from 1 to c-1 do l[n(r,i),1]:=l[n(r-1,c),3] + i od;

#The last column links:

 for i from 1 to r-1 do l[n(i+1,c),6]:=l[n(i,c),3] od;

#The NE links:
 for ii from 2 to r do for i from 1 to c-1 do
  l[n(ii,i),6]:=3+l[n(ii,i-1),6] od od;

#The E links:
 for i from 1 to r do for j from 1 to c-1 do
  l[n(i,j+1),4]:=l[n(i,j),1] od od;

 for i from 1 to r-1 do for j from 1 to c-1 do
  l[n(i,j+c+1),5]:=l[n(i,j),2] od od;

   s := sqrt(3.)/(2.)
```

```
#Matrices of zeros:

 for i from 1 to M+1 do for j from 1 to L+1 do
  A[i][j]:=0 od od; B := Matrix(2*N, L)

 for ijk from 1 to r+c do l[0,ijk]:=0; l[ijk,0]:=0 od;

#The next loop assigns values to the matrix B,
#which is the force balance matrix of all the nodes.

for i from 1 to r do
 for j from 1 to c do
   nnh := n(i, j); #equation for horizontal component
   nnv := c*r+nnh; #equation for vetical component
   if l[nnh, 1] > 0 then B[nnh, l[nnh, 1]] := -1 end if;
   if l[nnh, 2] > 0 then B[nnh, l[nnh, 2]] := -.5 end if;
   if l[nnh, 3] > 0 then B[nnh, l[nnh, 3]] := .5 end if;
   if l[nnh, 4] > 0 then B[nnh, l[nnh, 4]] := 1 end if;
   if l[nnh, 5] > 0 then B[nnh, l[nnh, 5]] := .5 end if;
   if l[nnh, 6] > 0 then B[nnh, l[nnh, 6]] := -.5 end if;
   if l[nnh, 1] > 0 then B[nnv, l[nnh, 1]] := 0 end if;
   if l[nnh, 2] > 0 then B[nnv, l[nnh, 2]] := s end if;
   if l[nnh, 3] > 0 then B[nnv, l[nnh, 3]] := s end if;
   if l[nnh, 4] > 0 then B[nnv, l[nnh, 4]] := 0 end if;
   if l[nnh, 5] > 0 then B[nnv, l[nnh, 5]] := -s end if;
   if l[nnh, 6] > 0 then B[nnv, l[nnh, 6]] := -s end if;
  od;
 od;

#The fixed parameters:
```

```
 for i to r do u[n(i, 1)] := 0; v[n(i, 1)] := 0 od;

#Fixing the first-column nodes by deleting the rows of the
#coefficient matrix that represent these parameters:

 l1 := [seq(n(j, 1), j = 1 .. r), seq(n(k, 1)+N, k = 1 .. r)];
 num := numelems(l1);
 At := DeleteRow(B,[seq(n(j,1), j=1..r), seq(n(k,1)+N, k=1..r)]);
 A := Transpose(At);

#Spring constants matrix (letting all equal to 1) :

 C := IdentityMatrix(L, outputoptions = [shape = rectangular])

#Here starts the step-by-step adjustments of the spring-constants:

 cc[0] := 1; b0 := 1; b1 := 12; b2 := 1; b3 := 3; b4 := 2
 for j from 1 to b1 do b[j] := 3.8 end do;
 for jj from 1 to b2 do b[b1+jj] := 1.0 end do;
 b[14] := 5.9;
 for j from 1 to b3 do b[b0+b1+b2+j] := 1.1 end do;
 for ij from 1 to b4 do b[b0+b1+b2+b3+ij] := .9 end do;
 b[20] := .5; b[21] := .5; b[22] := .5;
 for i from 23 to 38 do b[i] := .5 end do;

 k := 38;

 for i from 1 to k do cc[i] := cc[i-1]-b[i]/L end do;

 C[1, 1] := C[1, 1]+b[1]; C[4, 4] := C[4, 4]+b[2];
 C[7, 7] := C[7, 7]+b[3]; C[10, 10] := C[10, 10]+b[4];
```

```
 C[13, 13] := C[13, 13]+b[5]; C[16, 16] := C[16, 16]+b[6];
 C[104, 104] := C[104, 104]+b[7]; C[105, 105] := C[105, 105]+b[8];
 C[106, 106] := C[106, 106]+b[9]; C[107, 107] := C[107, 107]+b[10];
 C[108, 108] := C[108, 108]+b[11]; C[109, 109] := C[109, 109]+b[12];
 C[2, 2] := C[2, 2]+b[13]; C[103, 103] := C[103, 103]+b[14];
 C[19, 19] := C[19, 19]+b[15]; C[100, 100] := C[100, 100]+b[16];
 C[110, 110] := C[110, 110]+b[17];
 C[22, 22] := C[22, 22]+b[18]; C[111, 111] := C[111, 111]+b[19];
 C[63, 63] := C[63, 63]+b[20];
 C[39, 39] := C[39, 39]+b[21]; C[112, 112] := C[112, 112]+b[22];
 for i from 26 to 27 do C[i, i] := C[i, i]+b[23] end do;
 C[61, 61] := C[61, 61]+b[25]; C[76, 76] := C[76, 76]+b[26];
 C[88, 88] := C[88, 88]+b[27]; C[80, 80] := C[80, 80]+b[28];
 C[51, 51] := C[51, 51]+b[29]; C[92, 92] := C[92, 92]+b[30];
 C[14, 14] := C[14, 14]+b[31]; C[46, 46] := C[46, 46]+b[32];
 C[11, 11] := C[11, 11]+b[33]; C[15, 15] := C[15, 15]+b[34];
 C[57, 57] := C[57, 57]+b[35]; C[12, 12] := C[12, 12]+b[36];
 C[91, 91] := C[91, 91]+b[37]; C[58, 58] := C[58, 58]+b[38];

#Here the stiffness adjusting ends.

#Trace of the stiffness tensor checks if the
#total mass is unchanged:

 Trace(C);

 K := At.C.A #This is the stiffness matrix.

#Zz is a list of all parameters, even those I set = 0:

 Zz := [seq(u[i], i = 1 .. N), seq(v[i], i = 1 .. N)];
```

```
#X gets rid of zeros in Zz, i.e., those parameters I fixed:

 X := subsop(seq(l1[i] = NULL, i = 1 .. num), Zz);

#Xx just converts a list, X, into a matrix, which will
#lead to easier calculations:

 Xx := Matrix(X);

#KK multiplies the stiffness matrix with a matrix of
#the unknown parameters:

 KK := K.Transpose(Xx);

#And now we need Xx and KK as lists, for easier calculations:

 Xy := convert(Xx, list);
 S := convert(KK, list);

#Setting all force equations equal to zero:

 for j from 1 to M do SS[j] := S[j] = 0 end do;

#Choose the parameter(s) you want to apply force to,
#and set that force equal to a desired agnitude.
#Negative is either down or left, depending which
#parameter you chose to apply the force to, while
#the positive is either up or to the right. For this
#experiment I chose the very last parameter, which is
#the y-direction of the last node, node 48.
```

```
 SS[M] := S[M] = -0.5e-1;

#Now we solve the system of equations with the
#introduction of force to that last node:

 solve([seq(SS[i], i = 1 .. M)], [seq(Xy[j], j = 1 .. M)]);
 SLN := op(1, solve([seq(SS[i], i=1..M)],[seq(Xy[j],j=1..M)]));

 with(ListTools);
 with(Statistics);

#We are finally ready to compute the deflections of every node:

 for i from 1 to M do uu[i] := eval(u[i], SLN) end do;
 for i from 1 to M do vv[i] := eval(v[i], SLN) end do;
 UU := [seq(eval(u[i], SLN), i = 1 .. N),
        seq(eval(v[j], SLN), j = 1 .. N)];

#U is the list of node-deflections, excluding those that
#are fixed, and are, therefore, always equal to zero:

 U := subsop(seq(l1[i] = NULL, i = 1 .. num), UU);

#The list U is converted to a matrix, which is then transposed.
#We do this to compute the link elongations in the next step:

 with(LinearAlgebra);
 Xb_t := Matrix(U);

#And here are the links elongtions, matrix E,
```

```
#the internal forces in the links (or bars), Y,
#and the force matrix, F:

 E := A.Xb;
 Y := C.E;
 F := At.Y;

#And now back to a list so we can pick the most
#comprassed and the most elongated links:

 El := convert(E, list);
 Max := [FindMaximalElement(El, position)];
 Min := [FindMinimalElement(El, position)];

#mx is the most elongated, while mn is the most comprassed link:

 mx := Max[2]; mn := Min[2];

#And now we finally check and record the vertical displacement
#of the last link, which changes depending on what and how many
#improvements we make:

 U[M];
```

APPENDIX F

GRAPHING THE DAMAGED CANTILEVER

```
#The Damaged Cantilever Code

restart;

with(LinearAlgebra);

#The number of row-links, r, and the number of column-links, c:

 r := 4;
 c := 12;

 printlevel := 3;

#Assign n(i,j), i for the row, j for the column:

 for i to r do for j to c do n:=proc(i, j)
 options operator, arrow;

 (i-1)*c+j end proc end do end do;

 for i to r do for j to c do print((i, j) = (i-1)*c+j)
         end do end do;

#The number of fixed directions:
```

```
 p := 8;

#N is the total number of nodes, while L is the
#total number of links

 N := c*r;
 L := (c-1)*r+(r-1)*c+2*(sum(kik, kik = 1 .. r-1))+
 (c-r-1)*(r-1);

#The number of degrees of freedom:

 M := 2*N-p;

#l[i,j] is the j-th link of the i-th node, where j is
#one of 6 possible directions
#(E=1, SE=2, SW=3, W=4, NW=5 NE=6).

 l[n(1, c), 5] := 3*(c-1)+1 #The first link in the last column.

#There are no NW & NE links at the top (the fist column):

 for jj to c do for ii from 5 to 6 do l[n(1, jj), ii] := 0
         end do end do;

#There are no W & NW links in the first column:

 for k to r do for kk from 4 to 5 do l[n(k, 1), kk] := 0
         end do end do;

#There are no SE & SW links in the last row:
```

```
 for ji to c do for jij from 2 to 3 do l[n(r, ji), jij] := 0
         end do end do;

#There are no E & SE links in the last column:

 for i to r do for j to 2 do l[n(i, c), j] := 0
         end do end do;

#I define these to be :=0 just so I can use
#the next nested loop:

 for i to 6 do for j to 6 do l[n(0, i), j] := 0
         end do end do;

#These are the W & SW links of the last column;
#preparation for the big loop:

 for k to r-1 do for kk from 3 to 4 do
  l[n(k, c), kk] := k*((c-1)*3+1)-3*(kk-3)
          end do end do;

#This big nested loop takes care of E, SE, and SW links:

 for ii to r-1 do for i to c-1 do for j to 3 do
  l[n(ii, i), j] := j+l[n(ii, i-1), 3]
          end do end do end do;

#The last row links:

 for i to c-1 do l[n(r, i), 1] := l[n(r-1, c), 3]+i
         end do;
```

```
 for i to r-1 do l[n(i+1, c), 6] := l[n(i, c), 3]
         end do;

#The NE links:

 for ii from 2 to r do for i to c-1 do
  l[n(ii, i), 6] := 3+l[n(ii, i-1), 6] end do end do;

#The E links:

 for i to r do for j to c-1 do
  l[n(i, j+1), 4] := l[n(i, j), 1] end do end do;

 for i to r-1 do for j to c-1 do
  l[n(i, j+c+1), 5] := l[n(i, j), 2] end do end do;

 s := sqrt(3.)/(2.);

 for i to M+1 do for j to L+1 do A[i][j] := 0
         end do end do;

#Makes big matrices visable:

 interface(rtablesize = 100);

 B := Matrix(2*N, L);

 for ijk to r+c do l[0, ijk] := 0; l[ijk, 0] := 0 end do

#The next loop assigns values to the matrix B,
#which is the force balance matrix of all the nodes:
```

```
 for i to r do
  for j to c do
   nnh := n(i, j); #equation for horizontal component
   nnv := c*r+nnh; #equation for vetical component
   if l[nnh, 1] > 0 then B[nnh, l[nnh, 1]] := -1 end if;
   if l[nnh, 2] > 0 then B[nnh, l[nnh, 2]] := -1/2 end if;
   if l[nnh, 3] > 0 then B[nnh, l[nnh, 3]] := 1/2 end if;
   if l[nnh, 4] > 0 then B[nnh, l[nnh, 4]] := 1 end if;
   if l[nnh, 5] > 0 then B[nnh, l[nnh, 5]] := 1/2 end if;
   if l[nnh, 6] > 0 then B[nnh, l[nnh, 6]] := -1/2 end if;
   if l[nnh, 1] > 0 then B[nnv, l[nnh, 1]] := 0 end if;
   if l[nnh, 2] > 0 then B[nnv, l[nnh, 2]] := s end if;
   if l[nnh, 3] > 0 then B[nnv, l[nnh, 3]] := s end if;
   if l[nnh, 4] > 0 then B[nnv, l[nnh, 4]] := 0 end if;
   if l[nnh, 5] > 0 then B[nnv, l[nnh, 5]] := -s end if;
   if l[nnh, 6] > 0 then B[nnv, l[nnh, 6]] := -s end if
  end do
 end do

#The fixed parameters:

  for i to r do u[n(i, 1)] := 0; v[n(i, 1)] := 0 od;

#Fixing the first-column nodes by deleting the rows of the
#coefficient matrix that represent these parameters:

  l1 := [seq(n(j, 1), j = 1 .. r), seq(n(k, 1)+N,
  k = 1 .. r)];

  num := numelems(l1);
```

```
  At := DeleteRow(B,[seq(n(j,1), j=1..r), seq(n(k,1)+N,
  k=1..r)]);

  A := Transpose(At);

#Spring constants matrix (letting all equal to 1) :

  C := IdentityMatrix(L, outputoptions=[shape=rectangular])

#Here I change the stiffness of those links
#I want to strengthen or weaken:

 bb := 0.1e-1;

 C[1, 1] := bb; C[4, 4] := bb; C[38, 38] := bb;
 C[75, 75] := bb; C[43, 43] := bb;

 K := At.C.A; #This is the stiffness matrix.

#Zz is a list of all parameters, even those I set = 0:

 Zz := [seq(u[i], i = 1 .. N), seq(v[i], i = 1 .. N)];

#X gets rid of zeros in Zz, i.e., those parameters I fixed:

 X := subsop(seq(l1[i] = NULL, i = 1 .. num), Zz);

#Xx just converts a list, X, into a matrix, which will
#lead to easier calculations:

 Xx := Matrix(X);
```

```
#KK multiplies the stiffness matrix with a matrix of
#the unknown parameters:

 KK := K.Transpose(Xx);

#And now we need Xx and KK as lists, for easier calculations:

 Xy := convert(Xx, list);
 S := convert(KK, list);

#Setting all force equations equal to zero:

 for j from 1 to M do SS[j] := S[j] = 0 end do;

#Choose the parameter(s) you want to apply force to,
#and set that force equal to a desired agnitude.
#Negative is either down or left, depending which
#parameter you chose to apply the force to, while
#the positive is either up or to the right. For this
#experiment I chose the very last parameter, which is
#the y-direction of the last node, node 48.

 SS[M] := S[M] = -0.5e-1;

#Now we solve the system of equations with the
#introduction of force to that last node:

 solve([seq(SS[i], i = 1 .. M)], [seq(Xy[j], j = 1 .. M)]);
 SLN := op(1, solve([seq(SS[i], i=1..M)],
         [seq(Xy[j],j=1..M)]));
```

```
 with(ListTools);
 with(Statistics);

 for i from 1 to M do uu[i] := eval(u[i], SLN) end do;
 for i from 1 to M do vv[i] := eval(v[i], SLN) end do;

#This is where the accurate computation stops at the
#expense of graphing the damage of the cantilever.
#The computation continues in the next appendix.

#Graphing:

 with(plots);

 for i from 0 to c-1 do for j from 0 to r-1 do
  uuu[i, j] := i-(1/2)*j+eee*uu[c*j+i+1]; vvv[i, j] :=
  eee*vv[c*j+i+1]-j*s end do end do;

 display([seq(seq(plot([[uuu[i, j], vvv[i, j]],
         [uuu[i, j+1], vvv[i, j+1]]],
 scaling = constrained), i = 0 .. c-1), j = 0 .. r-2),
 seq(seq(plot([[uuu[i, j], vvv[i, j]], [uuu[i+1, j],
         vvv[i+1, j]]],
 scaling = constrained), i = 0 .. c-2), j = 0 .. r-1),
 seq(seq(plot([[uuu[i, j], vvv[i, j]], [uuu[i+1, j+1],
         vvv[i+1, j+1]]],
 scaling = constrained), i = 0 .. c-2), j = 0 .. r-2)]);

 display([seq(seq(plot([[i-(1/2)*j, -s*j],
         [i-(1/2)*j+1, -s*j]],
```

```
scaling = constrained), i = 0 .. c-2), j = 0 .. r-1),
seq(seq(plot([[i-(1/2)*j, -s*j],
        [i-(1/2)*j-1/2, -j*s-s]],
scaling = constrained), i = 0 .. c-1), j = 0 .. r-2),
seq(seq(plot([[i-(1/2)*j, -s*j],
        [i-(1/2)*j+1/2, -j*s-s]],
scaling = constrained), i = 0 .. c-2), j = 0 .. r-2)],
[seq(seq(plot([[uuu[i, j], vvv[i, j]],
        [uuu[i, j+1], vvv[i, j+1]]],
scaling = constrained, color = azure), i = 0 .. c-1),
        j = 0 .. r-2),
seq(seq(plot([[uuu[i, j], vvv[i, j]], [uuu[i+1, j],
        vvv[i+1, j]]],
scaling = constrained, color = blue), i = 0 .. c-2),
        j = 0 .. r-1),
seq(seq(plot([[uuu[i, j], vvv[i, j]], [uuu[i+1, j+1],
        vvv[i+1, j+1]]],
scaling = constrained, color = navy), i = 0 .. c-2),
        j = 0 .. r-2)]);

[seq(seq(print([[i, j], [i, j+1],
        sqrt((uuu[i, j]-uuu[i, j+1])^2+
(vvv[i, j]-vvv[i, j+1])^2)-1]), i = 0 .. c-1),
        j = 0 .. r-2),
seq(seq(print([[i, j], [i+1, j],
        sqrt((uuu[i, j]-uuu[i+1, j])^2+
(vvv[i, j]-vvv[i+1, j])^2)-1]), i = 0 .. c-2),
        j = 0 .. r-1),
seq(seq(print([[i, j], [i+1, j+1],
        sqrt((uuu[i, j]-uuu[i+1, j+1])^2+
(vvv[i, j]-vvv[i+1, j+1])^2)-1]), i = 0 .. c-2),
```

```
        j = 0 .. r-2)];

[seq(seq(print([[i, j], [i, j+1],
        sqrt((uuu[i, j]-uuu[i, j+1])^2+
(vvv[i, j]-vvv[i, j+1])^2)-1]), i = 0 .. c-1),
        j = 0 .. r-2),
seq(seq(print([[i, j], [i+1, j],
        sqrt((uuu[i, j]-uuu[i+1, j])^2+
(vvv[i, j]-vvv[i+1, j])^2)-1]), i = 0 .. c-2),
        j = 0 .. r-1),
seq(seq(print([[i, j], [i+1, j+1],
        sqrt((uuu[i, j]-uuu[i+1, j+1])^2+
(vvv[i, j]-vvv[i+1, j+1])^2)-1]), i = 0 .. c-2),
        j = 0 .. r-2)];

with(ListTools);

for i from 0 to c-1 do for j from 0 to r-2 do
 lk[n(j+1, i+1), n(j+2, i+1)] :=
         sqrt((uuu[i, j]-uuu[i, j+1])^2+
  (vvv[i, j]-vvv[i, j+1])^2)-1 end do end do;

for i from 0 to c-2 do for j from 0 to r-1 do
 lk[n(j+1, i+1), n(j+1, i+2)] :=
         sqrt((uuu[i, j]-uuu[i+1, j])^2+
  (vvv[i, j]-vvv[i+1, j])^2)-1 end do end do;

for i from 0 to c-2 do for j from 0 to r-2 do
 lk[n(j+1, i+1), n(j+2, i+2)] :=
         sqrt((uuu[i, j]-uuu[i+1, j+1])^2+
  (vvv[i, j]-vvv[i+1, j+1])^2)-1 end do end do;
```

```
 with(Statistics);

#Here I assign a low value to the link we damage,
#so we don't get it as the 'maximal element' again:

 lk[1, 2] := 0; lk[2, 3] := 0; lk[14, 15] := 0;
 lk[27, 28] := 0; lk[15, 27] := 0;

 lst_a := [seq(seq(n(j+1, i+1), j = 0 .. r-2), i = 0 .. c-1),
        seq(seq(n(j+1, i+1), j = 0 .. r-1), i = 0 .. c-2),
        seq(seq(n(j+1, i+1), j = 0 .. r-2), i = 0 .. c-2)];

 lst_b := [seq(seq(n(j+2, i+1), j = 0 .. r-2), i = 0 .. c-1),
        seq(seq(n(j+1, i+2), j = 0 .. r-1), i = 0 .. c-2),
        seq(seq(n(j+2, i+2), j = 0 .. r-2), i = 0 .. c-2)];

 lst_1 := [seq(seq(lk[n(j+1, i+1), n(j+2, i+1)], j = 0 .. r-2),
        i = 0 .. c-1), seq(seq(lk[n(j+1, i+1), n(j+1, i+2)],
        j = 0 .. r-1), i = 0 .. c-2), seq(seq(lk[n(j+1, i+1),
        n(j+2, i+2)], j = 0 .. r-2), i = 0 .. c-2)];

 zz := [FindMaximalElement(lst_1, position)]; kp := zz[2];

 lkk[lst_a[kp], lst_b[kp]] := lk[lst_a[kp], lst_b[kp]];

 for i to N do ll[i, i+1] := l[i, 1];
         ll[i, i-1] := l[i-1, 1] end do;

 for j to N do ll[j, j+c] := l[j, 3];
         ll[j, j+c+1] := l[j, 2] end do;
```

```
ll[lst_a[kp], lst_b[kp]];

rz := proc (s) options operator, arrow; .3*s/(0.1e-1+abs(s))+.3
end proc;

plot(rz(x), x = -.1 .. .1);
       0.3 s
s -> ---------- + 0.3
     0.01 + |s|

display(seq(plot([[(i-10)*(1/100), -10], [(i-10)*(1/100), 10]],
color = COLOR(HUE, rz((i-10)*(1/100)))), i = 1 .. 20),
axes = normal);

[seq(cspread[i], i = 1 .. 20)];

cspread := EvenSpread(Color("HSV", "Lime"), 20);

f := proc (x, y) options operator, arrow;
       7*sin((1/2)*x)+4*sin((1/4)*y) end proc

TH := .5; TD := .41; display([seq(seq(plot([[uuu[i, j],
vvv[i, j]],[uuu[i, j+1], vvv[i, j+1]]],
color = COLOR(HUE, rz(sqrt((uuu[i, j]-
 uuu[i, j+1])^2+(vvv[i, j]-vvv[i, j+1])^2)-1)),
 scaling = constrained),
  i = 0 .. c-1), j = 0 .. r-2), seq(seq(plot([[uuu[i, j],
  vvv[i, j]],
  [uuu[i+1, j], vvv[i+1, j]]], color = COLOR(HUE,
  rz(sqrt((uuu[i, j]-uuu[i+1, j])^2+(vvv[i, j]-
```

```
   vvv[i+1, j])^2)-1)), scaling = constrained),
   i = 0 .. c-2), j = 0 .. r-1), seq(seq(plot([[uuu[i, j],
   vvv[i, j]], [uuu[i+1, j+1], vvv[i+1, j+1]]],
   color = COLOR(HUE, rz(sqrt((uuu[i, j]-uuu[i+1, j+1])^2+
   (vvv[i, j]-vvv[i+1, j+1])^2)-1)), scaling = constrained),
   i = 0 .. c-2), j = 0 .. r-2)]);

#The link to break next:

 ll[lst_a[kp], lst_b[kp]];
```

APPENDIX G

THE COMPUTATION FOR THE DAMAGED CANTILEVER

```
#This is the continuation of the computation that was replaced
#by graphing in the previous appendix.

        UU := [seq(eval(u[i], SLN), i = 1 .. N),
         seq(eval(v[j], SLN), j = 1 .. N)];
        U := subsop(seq(l1[i] = NULL, i = 1 .. num), UU);
        with(LinearAlgebra);
        Xb_t := Matrix(U);
        Xb := Transpose(Xb_t);

#These are the elongations of every link; this is the list from
#which we're going to choose the most stretched and the most
#compressed link:

        E := A.Xb;

        Y := C.E; #These are the internal forces in the bars.

        F := At.Y; simplify(F); #The force matrix.

        El := convert(E, list);
```

```
        Ell := Array(El);

#Assign the most stretched (and/or the most compressed) link's
#elongation to 0 after breakage to avoid having it as the most
#stretched and/or compressed link again:

        Ell[1] := 0; Ell[38] := 0; Ell[39] := 0;
        Ell[72] := 0; Ell[73] := 0;

        El := convert(Ell, list);

        Max := [FindMaximalElement(El, position)];
        Min := [FindMinimalElement(El, position)];

#These are the most and the least elongated links:

        mx := Max[2]; mn := Min[2];

        for i to L do lk[i] := El[i] end do;

        lk[mx] := Max[1]; lk[mn] := Min[1];
```

APPENDIX H

THE FOURIER TRANSFORM METHOD IN 2-D

```
restart;
with(Student[LinearAlgebra]);
with(LinearAlgebra);
i := I;

A := i*(Matrix(3, 2, {(1, 1) = omega[1], (1, 2) = 0,
(2, 1) = 0, (2, 2) = omega[2], (3, 1) = (1/2)*omega[2],
(3, 2) = (1/2)*omega[1]}));

At := Transpose(A);

B := A.(1/(At.A)).At;

I3 := IdentityMatrix(3);

Ph := I3-B; #This is P-hat from the main text.

AAt := Determinant(At.A);

P := AAt.Ph; #P from the main text.

EE := Vector(3, {(1) = epsilon[11], (2) = epsilon[22],
(3) = epsilon[12]});
```

```
E := convert(EE, matrix);

P2 := P.E;

Rank(P2);                                   1

#Up until now everything in the code is the way it was
#depicted in the text. Because the expression was too
#complicated, the next line simplifies it, which results
#in identifying the wanted condition.

P3 := simplify(P2/(omega[1]^4+4*omega[1]^2*omega[2]^2+
omega[2]^4));

eval(P3, epsilon[11]*omega[2]^2-2*epsilon[12]*omega[1]*
omega[2]+epsilon[22]*omega[1]^2 = 0);

Matrix(3, 1, {(1, 1) = 0, (2, 1) = 0, (3, 1) = 0})

#And we get that P3=0, which implies that P2=0, if:

epsilon[11]*omega[2]^2-2*epsilon[12]*omega[1]*omega[2]+
epsilon[22]*omega[1]^2 = 0;
```

APPENDIX I

THE FOURIER TRANSFORM METHOD IN 3-D

```
restart;
with(LinearAlgebra);

A := Matrix(6, 3, {(1, 1) = omega[1], (1, 2) = 0, (1, 3) = 0,
(2, 1) = 0, (2, 2) = omega[2], (2, 3) = 0, (3, 1) = 0,
(3, 2) = 0, (3, 3) = omega[3], (4, 1) = (1/2)*omega[2],
(4, 2) = (1/2)*omega[1], (4, 3) = 0, (5, 1) = (1/2)*omega[3],
(5, 2) = 0, (5, 3) = (1/2)*omega[1], (6, 1) = 0,
(6, 2) = (1/2)*omega[3], (6, 3) = (1/2)*omega[2]});

B := A.(1/(Transpose(A).A)).Transpose(A);
E := Determinant(Transpose(A).A);
B2 := E*(IdentityMatrix(6)-B);
B3 := simplify((omega[1]^6+5*omega[1]^4*omega[2]^2+5*omega[1]^4*
omega[3]^2+5*omega[1]^2*omega[2]^4+17*omega[1]^2*omega[2]^2*
omega[3]^2+5*omega[1]^2*omega[3]^4+omega[2]^6+5*omega[2]^4*
omega[3]^2+5*omega[2]^2*omega[3]^4+omega[3]^6)*B2);

B4 := Vector[row](6, {(1) = epsilon[11], (2) = epsilon[22],
(3) = epsilon[33], (4) = epsilon[12], (5) = epsilon[13],
(6) = epsilon[23]});

B5 := Transpose(B4);
```

```
C := B3.B5;

C1 := simplify(16*C); v1 := C1[1]; v2 := C1[2];
v3 := C1[3]; v4 := C1[4]; v5 := C1[5]; v6 := C1[6];

solve({v1, v2, v3, v4, v5, v6}, [epsilon[11], epsilon[22],
epsilon[33], epsilon[12], epsilon[13], epsilon[23]]);

solve({v1, v2, v3, v4, v5, v6},
 [epsilon[12], epsilon[23], epsilon[33]]);

solve({v1, v2, v3, v4, v5, v6},
 [epsilon[11], epsilon[22], epsilon[23]]);

epsilon[22, 11] := 2*epsilon[12, 12]-epsilon[11, 22];
epsilon[23, 11] := -epsilon[11, 23]+epsilon[12, 13]+epsilon[13, 12];
epsilon[33, 22] := -epsilon[22, 33]+2*epsilon[23, 23];
epsilon[13, 22] := epsilon[12, 23]-epsilon[22, 13]+epsilon[23, 12];
epsilon[23, 13] := epsilon[12, 33]-epsilon[13, 23]+epsilon[33, 12];
epsilon[33, 11] := -epsilon[11, 33]+2*epsilon[13, 13];

#Now we check if, given the above definitions for the strains,
#the six 3-D compatiblity conditions hold:

cc1 := 2*epsilon[23, 23]-epsilon[22, 33]-epsilon[33, 22];
                                0
cc2 := 2*epsilon[13, 13]-epsilon[33, 11]-epsilon[11, 33];
                                0
cc3 := 2*epsilon[12, 12]-epsilon[11, 22]-epsilon[22, 11];
                                0
```

```
cc4 := epsilon[33, 12]+epsilon[12, 33]-epsilon[23, 13]-
       epsilon[13, 23];
                                0
cc5 := epsilon[11, 23]+epsilon[23, 11]-epsilon[13, 12]-
       epsilon[12, 13];
                                0
cc6 := epsilon[22, 13]+epsilon[13, 22]-epsilon[12, 23]-
       epsilon[23, 12];
                                0
#Each of the six compatibility conditions equals zero.
```

APPENDIX J

COMPATIBILITY CONDITIONS OF CUBOCTAHEDRON

```
restart;

with(ColorTools)
with(RandomTools);

#The nodes' coordinates of a cuboctahedron:

a[0, 1] := 0; a[0, 2] := 0; a[0, 3] := 0;
a[1, 1] := 1; a[1, 2] := 0; a[1, 3] := 0;
a[2, 1] := 1/2; a[2, 2] := (1/2)*sqrt(3); a[2, 3] := 0;
a[3, 1] := -1/2; a[3, 2] := (1/2)*sqrt(3); a[3, 3] := 0;
a[4, 1] := -1; a[4, 2] := 0; a[4, 3] := 0;
a[5, 1] := -1/2; a[5, 2] := -(1/2)*sqrt(3); a[5, 3] := 0;
a[6, 1] := 1/2; a[6, 2] := -(1/2)*sqrt(3); a[6, 3] := 0;
a[7, 1] := 1/2; a[7, 2] := -(1/6)*sqrt(3); a[7, 3] := (1/3)*sqrt(6);
a[8, 1] := -1/2; a[8, 2] := -(1/6)*sqrt(3);
                 a[8, 3] := (1/3)*sqrt(6);
a[9, 1] := 0; a[9, 2] := (1/3)*sqrt(3);
                 a[9, 3] := (1/3)*sqrt(6);
a[10, 1] := 1/2; a[10, 2] := (1/6)*sqrt(3);
                 a[10, 3] := -(1/3)*sqrt(6);
a[11, 1] := -1/2; a[11, 2] := (1/6)*sqrt(3);
                 a[11, 3] := -(1/3)*sqrt(6);
```

```
a[12, 1] := 0; a[12, 2] := -(1/3)*sqrt(3);
                a[12, 3] := -(1/3)*sqrt(6);

#Normal distribution to generate small random perturbation of a
#regular cuboctahedron is written next. This part is commented out
#if one wants to see what the compatibility conditions of an
#unperturbed cuboctahedron are.

sigma := 0.5e-1;

for ii to 12 do for jj to 3 do
        a[ii, jj] := evalf(a[ii,jj])+
        Generate(distribution(Normal(0, sigma)))
end do end do;

#The 13 (perturbed) nodes' coordinates:

for i from 0 to 12 do V[i] := [a[i, 1], a[i, 2], a[i, 3]] end do;

#The connected nodes:

EI := [seq([i, 0], i = 1 .. 12), [1, 2], [1, 6],
[1, 7], [1, 10], [2, 3], [2, 9], [2, 10], [3, 4],
[3, 9], [3, 11], [4, 5], [4, 8], [4, 11], [5, 6],
[5, 8], [5, 12], [6, 7], [6, 12], [7, 8], [7, 9],
[8, 9], [10, 11], [10, 12], [11, 12]];

#The links:

for i to 36 do E[op(EI[i])] := [V[EI[i][1]], V[EI[i][2]]] end do;
```

```
#Fixing the total angular momentum:

f1 := sum(a[ik, 1]*u[ik, 2]-a[ik, 2]*u[ik, 1], ik = 1 .. 12) = 0;
f2 := sum(a[ik, 2]*u[ik, 3]-a[ik, 3]*u[ik, 2], ik = 1 .. 12) = 0;
f3 := sum(-a[ik, 1]*u[ik, 3]+a[ik, 3]*u[ik, 1], ik = 1 .. 12) = 0;

#The links written as a vectors:

for k from 0 to 12 do for m from 0 to 12 do
l[k, m][1] := a[k, 1]-a[m, 1] end do end do;

for n from 0 to 12 do for p from 0 to 12 do
l[n, p][2] := a[n, 2]-a[p, 2] end do end do;

for r from 0 to 12 do for q from 0 to 12 do
l[r, q][3] := a[r, 3]-a[q, 3] end do end do;

#In order to detect the infinitesimal flexibility of a nongeneric
#cuboctahedron we need to set each elongation to zero, and comment
#out the perturbation. This is how nonrigid motion can be detected.

for ij from 0 to 12 do for ji from 0 to 12 do
lambda[ij, ji] := 0 end do end do

#The following is the main system of equations:

z[1] := l[0, 1][1]*u[1][1]+l[0, 1][2]*u[1][2]+
 l[0, 1][3]*u[1][3] = lambda[0, 1];
z[2] := l[0, 2][1]*u[2][1]+l[0, 2][2]*u[2][2]+
 l[0, 2][3]*u[2][3] = lambda[0, 2];
z[4] := l[0, 4][1]*u[4][1]+l[0, 4][2]*u[4][2]+
```

```
 l[0, 4][3]*u[4][3] = lambda[0, 4];
z[5] := l[0, 5][1]*u[5][1]+l[0, 5][2]*u[5][2]+
 l[0, 5][3]*u[5][3] = lambda[0, 5];
z[3] := l[0, 3][1]*u[3][1]+l[0, 3][2]*u[3][2]+
 l[0, 3][3]*u[3][3] = lambda[0, 3];
z[6] := l[0, 6][1]*u[6][1]+l[0, 6][2]*u[6][2]+
 l[0, 6][3]*u[6][3] = lambda[0, 6];
z[7] := l[0, 7][1]*u[7][1]+l[0, 7][2]*u[7][2]+
 l[0, 7][3]*u[7][3] = lambda[0, 7];
z[8] := l[0, 8][1]*u[8][1]+l[0, 8][2]*u[8][2]+
 l[0, 8][3]*u[8][3] = lambda[0, 8];
z[9] := l[0, 9][1]*u[9][1]+l[0, 9][2]*u[9][2]+
 l[0, 9][3]*u[9][3] = lambda[0, 9];
z[10] := l[0, 10][1]*u[10][1]+l[0, 10][2]*u[10][2]+
 l[0, 10][3]*u[10][3] = lambda[0, 10];
z[11] := l[0, 11][1]*u[11][1]+l[0, 11][2]*u[11][2]+
 l[0, 11][3]*u[11][3] = lambda[0, 11];
z[12] := l[0, 12][1]*u[12][1]+l[0, 12][2]*u[12][2]+
 l[0, 12][3]*u[12][3] = lambda[0, 12];
z[13] := (u[1][1]-u[2][1])*l[1, 2][1]+
 (u[1][2]-u[2][2])*l[1, 2][2]+
  (u[1][3]-u[2][3])*l[1, 2][3] = lambda[1, 2];
z[14] := (u[1][1]-u[6][1])*l[1, 6][1]+
 (u[1][2]-u[6][2])*l[1, 6][2]+
  (u[1][3]-u[6][3])*l[1, 6][3] = lambda[1, 6];
z[15] := (u[1][1]-u[7][1])*l[1, 7][1]+
 (u[1][2]-u[7][2])*l[1, 7][2]+
  (u[1][3]-u[7][3])*l[1, 7][3] = lambda[1, 7];
z[16] := (u[1][1]-u[10][1])*l[1, 10][1]+
 (u[1][2]-u[10][2])*l[1, 10][2]+
  (u[1][3]-u[10][3])*l[1, 10][3] = lambda[1, 10];
```

```
z[17] := (u[2][1]-u[3][1])*l[2, 3][1]+
 (u[2][2]-u[3][2])*l[2, 3][2]+
  (u[2][3]-u[3][3])*l[2, 3][3] = lambda[2, 3];
z[18] := (u[2][1]-u[7][1])*l[2, 7][1]+
 (u[2][2]-u[7][2])*l[2, 7][2]+
  (u[2][3]-u[7][3])*l[2, 7][3] = lambda[2, 7];
z[19] := (u[2][1]-u[11][1])*l[2, 11][1]+
 (u[2][2]-u[11][2])*l[2, 11][2]+
  (u[2][3]-u[11][3])*l[2, 11][3] = lambda[2, 11];
z[20] := (u[3][1]-u[4][1])*l[3, 4][1]+
 (u[3][2]-u[4][2])*l[3, 4][2]+
  (u[3][3]-u[4][3])*l[3, 4][3] = lambda[3, 4];
z[21] := (u[3][1]-u[8][1])*l[3, 8][1]+
 (u[3][2]-u[8][2])*l[3, 8][2]+
  (u[3][3]-u[8][3])*l[3, 8][3] = lambda[3, 8];
z[22] := (u[3][1]-u[11][1])*l[3, 11][1]+
 (u[3][2]-u[11][2])*l[3, 11][2]+
  (u[3][3]-u[11][3])*l[3, 11][3] = lambda[3, 11];
z[23] := (u[4][1]-u[5][1])*l[4, 5][1]+
 (u[4][2]-u[5][2])*l[4, 5][2]+
  (u[4][3]-u[5][3])*l[4, 5][3] = lambda[4, 5];
z[24] := (u[4][1]-u[8][1])*l[4, 8][1]+
 (u[4][2]-u[8][2])*l[4, 8][2]+
  (u[4][3]-u[8][3])*l[4, 8][3] = lambda[4, 8];
z[25] := (u[4][1]-u[12][1])*l[4, 12][1]+
 (u[4][2]-u[12][2])*l[4, 12][2]+
  (u[4][3]-u[12][3])*l[4, 12][3] = lambda[4, 12];
z[26] := (u[5][1]-u[6][1])*l[5, 6][1]+
 (u[5][2]-u[6][2])*l[5, 6][2]+
  (u[5][3]-u[6][3])*l[5, 6][3] = lambda[5, 6];
z[27] := (u[5][1]-u[9][1])*l[5, 9][1]+
```

```
 (u[5][2]-u[9][2])*l[5, 9][2]+
  (u[5][3]-u[9][3])*l[5, 9][3] = lambda[5, 9];
z[28] := (u[5][1]-u[12][1])*l[5, 12][1]+
 (u[5][2]-u[12][2])*l[5, 12][2]+
  (u[5][3]-u[12][3])*l[5, 12][3] = lambda[5, 12];
z[29] := (u[6][1]-u[9][1])*l[6, 9][1]+
 (u[6][2]-u[9][2])*l[6, 9][2]+
  (u[6][3]-u[9][3])*l[6, 9][3] = lambda[6, 9];
z[30] := (u[6][1]-u[10][1])*l[6, 10][1]+
 (u[6][2]-u[10][2])*l[6, 10][2]+
  (u[6][3]-u[10][3])*l[6, 10][3] = lambda[6, 10];
z[31] := (u[7][1]-u[8][1])*l[7, 8][1]+
 (u[7][2]-u[8][2])*l[7, 8][2]+
  (u[7][3]-u[8][3])*l[7, 8][3] = lambda[7, 8];
z[32] := (u[7][1]-u[9][1])*l[7, 9][1]+
 (u[7][2]-u[9][2])*l[7, 9][2]+
  (u[7][3]-u[9][3])*l[7, 9][3] = lambda[7, 9];
z[33] := (u[8][1]-u[9][1])*l[8, 9][1]+
 (u[8][2]-u[9][2])*l[8, 9][2]+
  (u[8][3]-u[9][3])*l[8, 9][3] = lambda[8, 9];
z[34] := (u[10][1]-u[11][1])*l[10, 11][1]+
 (u[10][2]-u[11][2])*l[10, 11][2]+
  (u[10][3]-u[11][3])*l[10, 11][3] = lambda[10, 11];
z[35] := (u[10][1]-u[12][1])*l[10, 12][1]+
 (u[10][2]-u[12][2])*l[10, 12][2]+
  (u[10][3]-u[12][3])*l[10, 12][3] = lambda[10, 12];
z[36] := (u[11][1]-u[12][1])*l[11, 12][1]+
 (u[11][2]-u[12][2])*l[11, 12][2]+
  (u[11][3]-u[12][3])*l[11, 12][3] = lambda[11, 12];

l1 := f1, f2, f3;
```

```
l2 := seq(z[i], i = 1 .. 36);

#The following command solves for the unknown displacements,
#and computes the compatibility conditions:

eliminate({l1, l2}, {seq(seq(u[j][k], k = 1 .. 3), j = 1 .. 12)})
```

REFERENCES

[1] M. Braun. Compatibility conditions for discrete elastic structures. *Rendiconti del Seminario Matematico Università e Politecnico di Torino*, 58(1):37–48, 2000. URL ftp://ftp.math.ethz.ch/hg/EMIS/journals/RSMT/58-1/37.pdf.

[2] C. R. Calladine. Buckminster Fuller's 'tensegrity' structures and Clerk Maxwell's rules for the construction of stiff frames. *International Journal of Solids and Structures*, 14:161–172, 1978. CODEN IJSOAD. ISSN 0020-7683 (print), 1879-2146 (electronic). URL http://arxiv.org/pdf/1010.5440v3.pdf.

[3] A. Cherkaev and S. Leelavanichkul. Principle of optimization of structures against an impact. *Journal of Physics: Conference Series*, 319 (1):1–16, 2011. CODEN JPCSDZ. ISSN 1742-6588 (print), 1742-6596 (electronic).

[4] A. Cherkaev and S. Leelavanichkul. An impact protective structure with bistable links. *International Journal of Damage Mechanics*, 21(5):697–711, 2012. CODEN IDMEEH. ISSN 1056-7895 (print), 1530-7921 (electronic). URL http://ijd.sagepub.com/content/21/5/697.abstract.

[5] A. Cherkaev, V. Vinogradov, and S. Leelavanichkul. The waves of damage in elastic-plastic lattices with waiting links: Design and simulation. *Mechanics of Materials: an International Journal*, 38(8–10): 748–756, October 2006. CODEN MSMSD3. ISSN 0167-6636 (print), 1872-7743 (electronic).

[6] A. Cherkaev and L. Zhornitskaya. Protective structures with waiting links and their damage evolution. *Multibody System Dynamics*, 13 (1):53–67, February 2005. ISSN 1384-5640 (print), 1573-272X (electronic). URL http://link.springer.com/article/10.1007/s11044-005-5166-z.

[7] Andrej Cherkaev, Andrei Kouznetsov, and Alexander Panchenko. Still states of bistable lattices, compatibility, and phase transition. *Continuum Mechanics and Thermodynamics*, 22(6-8):421–444, 2010. CODEN CMETEJ. ISSN 0935-1175 (print), 1432-0959 (electronic). URL http://dx.doi.org/10.1007/s00161-010-0161-x.

[8] Andrej Cherkaev and Liya Zhornitskaya. Dynamics of damage in two-dimensional structures with waiting links. In A. B. Movchan, editor, *IUTAM Symposium on Asymptotics, Singularities*

and Homogenisation in Problems of Mechanics: proceedings of the IUTAM symposium held in Liverpool, United Kingdom, 8–11 July 2002, volume 113 of *Solid Mechanics and Its Applications*, pages 273–283. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2003. ISBN 1-4020-1780-4 (print), 1-4020-2604-8 (e-book). LCCN QA808.2 .I98 2002. URL http://www.loc.gov/catdir/enhancements/fy0823/2003063756-d.html; http://www.loc.gov/catdir/enhancements/fy0823/2003063756-t.html.

[9] Pei Chi Chou and Nicholas J. Pagano. *Elasticity: Tensor, Dyadic, and Engineering Approaches*. Dover books on engineering. Dover Publications, Inc., New York, NY, USA, 1992. ISBN 0-486-66958-0. xiv + 290 pp. LCCN QA931 .C5 1992. US$8.95.

[10] Mykyta V. Chubynsky. *Rigidity and Self-Organization of Random Networks*. Ph.D. thesis, Michigan State University, East Lansing, MI, USA, 2003. URL http://search.proquest.com/pqdtglobal/docview/305328707/.

[11] R. Connelly, P. W. Fowler, S. D. Guest, B. Schulze, and W. J. Whiteley. When is a symmetric pin-jointed framework isostatic?, September 2008. URL http://arxiv.org/pdf/0803.2325.pdf.

[12] R. Connelly, T. Jordán, and W. Whiteley. Generic global rigidity of body-bar frameworks. *Journal of Combinatorial Theory (Series B)*, 103: 689–705, October 2013. CODEN JCBTB8. ISSN 0095-8956 (print), 1096-0902 (electronic). URL http://www.math.cornell.edu/~connelly/Connelly-Jordan-Whiteley.pdf.

[13] Robert Connelly. Conjectures and open questions in rigidity. In *Proceedings of the International Congress of Mathematicians (Helsinki, 1978)*, pages 407–414. Academiæ Scientiarum Fennicæ Mathematica, Helsinki, Finland, 1980. URL http://www.mathunion.org/ICM/ICM1978.1/Main/icm1978.1.0407.0414.ocr.pdf.

[14] Robert Connelly and M. Terrell. Tenségrités symétriques globalement rigides. (French) [Globally rigid symmetric tensegrities]. *Structural Topology*, 21:59–78, 1995. ISSN 0226-9171. URL http://upcommons.upc.edu/handle/2099/1099. Dual French-English text.

[15] Satyan L. Devadoss and Joseph O'Rourke. *Discrete and Computational Geometry*. Princeton University Press, Princeton, NJ, USA, 2011. ISBN 0-691-14553-9 (hardcover). xi + 255 pp. LCCN QA448.D38 D48 2011.

[16] H. Gluck. Almost all simply connected services are rigid. *Lecture Notes in Mathematics*, 438:225–239, August 2006. CODEN LNMAA2. ISSN 0075-8434 (print), 1617-9692 (electronic). URL http://www.math.cornell.edu/~web7510/Gluck-almost-all.pdf. The original paper is from 1973.

[17] S. D. Guest and P. W. Fowler. Symmetry-extended counting rules for periodic networks. Unpublished online notes., October 2013. URL http://www2.eng.cam.ac.uk/~sdg/preprint/periodic.pdf.

[18] Ivan Izmestiev. *Infinitesimal Rigidity of Frameworks and Surfaces*. Kyushu University, 744 Motooka, Nishi-Ku, Fukuoka, 819-0395, Japan, Spring 2009. URL http://page.mi.fu-berlin.de/izmestiev/Teaching/InfRig.pdf.

[19] Donald J. Jacobs and Bruce Hendrickson. An algorithm for two-dimensional rigidity percolation: The pebble game. *Journal of Computational Physics*, 137:346–365, 1997. CODEN JCTPAH. ISSN 0021-9991 (print), 1090-2716 (electronic). URL http://faculty.cs.tamu.edu/ajiang/PebbleGame.pdf.

[20] Piaras Kelly. Mechanics lecture notes. Web site with five on-line books of lecture notes on solid mechanics, continuum mechanics, and finite elements., 2015. URL http://homepages.engineering.auckland.ac.nz/~pkel015/SolidMechanicsBooks/index.html.

[21] Gerard Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 4(4):331–340, October 1970.

[22] András Lengyel. *Analogy between equilibrium of structures and compatibility of mechanisms*. Ph.D. thesis, University of Oxford, Oxford, UK, 2002. URL http://www.eng.ox.ac.uk/civil/publications/theses/lengyel.pdf.

[23] J. Clerk Maxwell. On the calculation of the equilibrium and stiffness of frames. *Philosophical Magazine*, 27:294–299, 1864. CODEN PHMAA4. ISSN 0031-8086.

[24] Igor Pak. Lectures on discrete and polyhedral geometry. Online book, in preparation., April 28, 2010. URL http://math.ucla.edu/~pak/book.htm.

[25] Ravi Mokashi Punekar and Avinash Shinde. Geometrical construction in three dimensional forms. URL http://www.dsource.in/course/geometry-design/introduction/index.html.

[26] David J. Raymond. Introduction to continuum mechanics, 1999. URL http://kestrel.nmt.edu/~raymond/classes/ph536/continuum.pdf. Online class notes; original edition 1994.

[27] E. Ross, B. Schulze, and W. Whiteley. Finite motions from periodic frameworks with added symmetry. *International Journal of Solids and Structures*, 48, 2011. CODEN IJSOAD. ISSN 0020-7683 (print), 1879-2146 (electronic). URL http://arxiv.org/pdf/1010.5440v3.pdf.

[28] Ben Roth. Questions on the rigidity of structures. *Structural Topology*, 4:67–71, 1980.

[29] Ben Roth. Unknown. Online book, in preparation., April 1987. URL http://www.math.cornell.edu/~connelly/rigidity.chapter.1.pdf.

[30] Martin Howard Sadd. *Elasticity: Theory, Applications, and Numerics*. Elsevier Butterworth Heinemann, Amsterdam, The Netherlands, 2005. ISBN 0-12-605811-3, 1-4175-4962-9 (e-book). xii + 461 pp. LCCN QA931 .S2 2005.

[31] Gilbert Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, Wellesley, MA, USA, 1986. ISBN 0-9614088-0-4. ix + 758 pp. LCCN QA37.2 .S87 1986. US$39.00 (est.).

[32] M. F. Thorpe and D. J. Jacobs. Generic rigidity percolation in two dimensions. Technical report, Department of Physics and Astronomy and Center for Fundamental Materials Research, Michigan State University, East Lansing, MI 48824, USA, 1996. URL https://www.pa.msu.edu/ftp/pub/thorpe/rigidity.pdf.